JC16 Rec'd PCT/PTO 26 DEC 2001

S&H Form: FORM PTO-1390 (2/01)

U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE **TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371**		ATTORNEY'S DOCKET NUMBER 1454.1128 10/019005
INTERNATIONAL APPLICATION NO. PCT/DE00/01764	INTERNATIONAL FILING DATE 30 May 2000	PRIORITY DATE CLAIMED 23 June 1999

TITLE OF INVENTION
ASSEMBLY, METHOD, COMPUTER PROGRAM AND STORAGE MEDIUM WHICH CAN BE COMPUTER-READ FOR THE COMPUTER-AIDED COMPENSATION OF A STATE OF INEQUILIBRIUM

APPLICANT(S) FOR DO/EO/US
Ralf NEUNEIER et al.

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. ☒ This is a FIRST submission of items concerning a filing under 35 U.S.C. 371.
2. ☒ This is an express request to immediately begin national examination procedures (35 U.S.C. 371(f)).
3. ☒ The US has been elected by the expiration of 19 months from the priority date (PCT Article 31).
4. ☒ A copy of the International Application as filed (35 U.S.C. 371(c)(2))
   a. ☒ is transmitted herewith (required only if not transmitted by the International Bureau).
   b. ☐ has been transmitted by the International Bureau.
   c. ☐ is not required, as the application was filed in the United States Receiving Office (RO/US).
5. ☒ A translation of the International Application into English (35 U.S.C. 371(c)(2)).
6. ☐ Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371(c)(3))
   a. ☐ are transmitted herewith (required only if not transmitted by the International Bureau).
   b. ☐ have been transmitted by the International Bureau.
   c. ☐ is not required, as the application was filed in the United States Receiving Office (RO/US)
7. ☐ A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
8. ☒ An oath or declaration of the inventor (35 U.S.C. 371(c)(4)).
9. ☐ A translation of the Annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 10-15 below concern document(s) or information included:

10. ☒ An Information Disclosure Statement Under 37 CFR 1.97 and 1.98.
11. ☒ An assignment document for recording.
    Please mail the recorded assignment document to:
    a. ☒ the person whose signature, name & address appears at the bottom of this document.
    b. ☐ the following:
12. ☒ A preliminary amendment.
13. ☒ A substitute specification
14. ☐ A change of power of attorney and/or address letter.
15. ☐ Other items or information:

Two CD-R's, each containing a copy of the program listing in the International Application, PCT EASY forms filed with International Application, copy of cover page of International Application as published, and copy of International Search Report.

10/019005

JC13 Rec'd PCT/PTO 26 DEC 2001

☒ The U.S. National Fee (35 U.S.C. 371(c)(1)) and other fees as follows:

CLAIMS	(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) CALCULATIONS
	TOTAL CLAIMS	18 -20=	0	x $ 18.00	0.00
	INDEPENDENT CLAIMS	3 -3=	0	x $ 84.00	0.00
	MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+$280.00	0.00
	BASIC NATIONAL FEE (37 CFR 1.492(a)(1)-(4): ☐ Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO ......$1,040 ☒ International preliminary examination fee (37 C.F.R. 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO.........$ 890 ☐ International preliminary examination fee (37 C.F.R. 1.482) not paid to USPTO but international search fee (37 C.F.R. 1.445(a)(2) paid to USPTO...$ 740 ☐ International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provision of PCT Article 33(1)-(4)...........$ 710 ☐ International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(2) to (4) ....$ 100				890.00
	Surcharge of $130 for furnishing the National fee or oath or declaration later than ☐ 20 ☐ 30 mos. from the earliest claimed priority date (37 CFR 1.482(e)).				0.00
		TOTAL OF ABOVE CALCULATIONS			890.00
	Reduction by 1/2 for filing by small entity, if applicable. Affidavit must be filed also. (Note 37 CFR 1.9, 1.27, 1.28.)				
		SUBTOTAL			890.00
	Processing fee of $130 for furnishing the English Translation later than [ ] 20 [ ] 30 mos. from the earliest claimed priority date (37 CFR 1.482(f)).				
		TOTAL NATIONAL FEE			0.00
	Fee for recording the enclosed assignment (37 CFR 1.21(h)).				+ 40.00
		TOTAL FEES ENCLOSED			930.00

a. ☒ A check in the amount of $ 930.00 to cover the above fees is enclosed.

b. ☐ Please charge my Deposit Account No. 19-3935 in the Amount of $ to cover the above fees. A duplicate copy of this sheet is enclosed.

c. ☒ The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account No. 19-3935. A duplicate copy of this sheet is enclosed.

21171

PATENT TRADEMARK OFFICE

SUBMITTED BY: STAAS & HALSEY LLP

Type Name	Richard A. Gollhofer	Reg. No.	31,106
Signature	Richard A. Gollhofer	Date	12/26/01

Docket No.: 1454.1128

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Ralf NEUNEIER et al.

Serial No. Group Art Unit:

Confirmation No.

Filed: (concurrently) Examiner:

For: SYSTEM FOR COMPENSATION OF A STATE OF INEQUILIBRIUM (as amended)

PRELIMINARY AMENDMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Before examination of the above-identified application, please amend the application as follows:

IN THE TITLE:

Please DELETE the Title in its entirety and REPLACE with the following new Title.

-- SYSTEM FOR COMPENSATION OF A STATE OF INEQUILIBRIUM --.

IN THE SPECIFICATION:

Please REPLACE the pending specification with the substitute specification attached hereto.

IN THE CLAIMS:

Please CANCEL without prejudice or disclaimer claims 1-19 in the underlying PCT application and ADD new claims 20-37 in accordance with the following:

20. (NEW) A computer system for compensation of an inequilibrium state of a first technical system, comprising:

a first neural network describing the first technical system; and

a second neural network describing a second technical system and coupled to the first neural network to compensate for an inequilibrium state of the first technical system.

21. (NEW) The arrangement as claimed in claim 20, wherein said first neural network has at least a first input computing element and a second output computing element.

22. (NEW) The arrangement as claimed in claim 21, wherein said second neural network has at least a second input computing element and a second output computing element.

23. (NEW) The arrangement as claimed in claim 22, wherein at least some of the computing elements are artificial neurons.

24. (NEW) The arrangement as claimed in claim 22, wherein at least some connections between computing elements are of variable configuration.

25. (NEW) The arrangement as claimed in claim 22, wherein at least some connections between computing elements have identical weighting values.

26. (NEW) The arrangement as claimed in claim 22, wherein the first technical system and the second technical system are identical.

27. (NEW) The arrangement as claimed in claim 22, wherein the first technical system and the second technical system are each subsystems of a common overall system.

28. (NEW) The arrangement as claimed in claim 27, wherein said first and second neural networks further determine a dynamic of the common overall system.

29. (NEW) The arrangement as claimed in claim 27, wherein said first and second neural networks further predict a future state of the common overall system.

30. (NEW) The arrangement as claimed in claim 29, wherein said first and second neural networks further performing at least one of monitoring and controlling the common overall system.

31. (NEW) The arrangement as claimed in claim 30, wherein the common overall system is a chemical reactor.

32. (NEW) A method for computer-supported compensation of an inequilibrium state of a first technical system, comprising:

supplying a first input variable to a first neural network describing the first technical system;

determining a first output variable, describing an inequilibrium state of the first technical system, for the first input variable using the first neural network;

supplying the first output variable, as a second input variable, to a second neural network which describes a second technical system; and

determining a second output variable, describing a state of the second technical system, for the second input variable using the second neural network, to compensate for the inequilibrium state of the first technical system by the second neural network.

33. (NEW) The method as claimed in claim 32, wherein the first technical system and the second technical system each describe a subsystem of a common overall system.

34. (NEW) The method as claimed in claim 33, further comprising determining a dynamic of the common overall system using the state of the second technical system.

35. (NEW) The method as claimed in one of claims 34, further comprising predicting a future state of the common overall system.

36. (NEW) The method as claimed in claim 35, further comprising at least one of monitoring and controlling the common overall system.

37. (NEW) A computer-readable medium storing a computer program for controlling a computer to perform a method for compensation of an inequilibrium state of a first technical system, said method comprising:

supplying a first input variable to a first neural network describing the first technical system;

determining a first output variable, describing an inequilibrium state of the first technical system, for the first input variable using the first neural network;

supplying the first output variable, as a second input variable, to a second neural network which describes a second technical system; and

determining a second output variable, describing a state of the second technical system, for the second input variable using the second neural network, to compensate for the inequilibrium state of the first technical system by the second neural network.

IN THE ABSTRACT:

Please DELETE the Abstract in its entirety and replace with the attached Substitute Abstract.

REMARKS

This Preliminary Amendment is submitted to improve the form of the English translation as filed. It is respectfully requested that this Preliminary Amendment be entered in the above-referenced application.

In accordance with the foregoing, claims 1-19 have been canceled and claims 20-37 have been added. Thus, claims 20-37 are pending and are under consideration.

A substitute specification is also being filed herewith. The substitute specification is accompanied by a marked-up copy of the original specification.

If there are any questions regarding these matters, such questions can be addressed by telephone to the undersigned. Otherwise, an early action on the merits is respectfully solicited.

If any further fees are required in connection with the filing of this Preliminary Amendment, please charge same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 12/26/01

By: Richard A. Gollhofer

Richard A. Gollhofer
Registration No. 31,106

700 Eleventh Street, NW, Suite 500
Washington, D.C. 20001
(202) 434-1500

10/019005
JC13 Rec'd PCT/PTO 26 DEC 2001

SUBSTITUTE SPECIFICATION

TITLE OF THE INVENTION

SYSTEM FOR COMPENSATION OF AN INEQUILIBRIUM STATE

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is based on and hereby claims priority to German Application No. 199 28 776.7 filed on June 23, 1999, the contents of which are hereby incorporated by reference.

REFERENCE TO COMPUTER PROGRAM LISTING, COMPACT DISC APPENDIX

[0002] A compact disc is included herewith and incorporated by reference herein having thereon a computer program listing appendix in the ASCII uncompressed text format with ASCII carriage return, ASCII line feed and all control codes defined in ASCII, having computer compatibility with IBM PC/XT/AT or compatibles, having operating system compatibility with MS-Windows and including file PROGRA~1 (ProgramListing.txt in Windows) of 98,120 bytes, created on December 14, 2001.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0003] The invention relates to an system, a method, a computer program event and a computer-readable storage medium for the computer-supported compensation of an inequilibrium state of a technical system.

2. Description of the Related Art

[0004] From S. Haykin, Neural Networks: a Comprehensive Foundation, McMillan College Publishing Company, 1994, pages 498-533, it is known to use a neural network to determine states of a dynamic system and a dynamic which is the basis for a dynamic system.

[0005] Generally, a dynamic process which occurs in a dynamic system is usually described by a state transition description which is not visible to an observer of the dynamic process, and a starting equation which describes observable variables of the technical dynamic process.

[0006] Such a structure is illustrated in Fig 7.

[0007] A dynamic system 700 is subject to the influence of an external input variable u with a predefinable dimension, an input variable ut at a point in time t being designated as ut:

$$u_t \in \Re^1,$$

l designating a natural number.

[0008] The input variable ut at a point in time t brings about a change in the dynamic process which occurs in the dynamic system 700.

[0009] An inner state st ($s_t \in \Re^m$) with a predefinable dimension m at a point in time t cannot be observed by an observer of the dynamic system 200.

[0010] A state transition of the inner state st of the dynamic process is brought about as a function of the inner state st and the input variable ut, and the state of the dynamic process changes into a subsequent state st+1 at a subsequent point in time t+1.

Here the following applies:

$$s_{t+1} = f(s_t, u_t). \quad (1)$$

where f(.) designates a general mapping rule.

[0011] An output variable y_t, which can be observed by an observer of the dynamic system 700, at a point in time t depends on the input variable u_t and on the inner state s_t.

[0012] The output variable y_t ($y_t \in \Re^n$) has a predefinable dimension n.

[0013] The dependence of the output variable y_t on the input variable u_t and the inner state s_t of the dynamic process is given by the following general rule:

$$y_t = g(s_t, u_t), \quad (2)$$

where g(.) designates a general mapping rule.

[0014] In Haykin, an arrangement of interconnected computing elements in the form of a neural network of interconnected neurons is used to describe the dynamic system 700. The

connections between the neurons of the neural network are weighted. The weightings of the neural network are combined in a parameter vector v.

[0015] An inner state of a dynamic system, which is subject to a dynamic process, thus depends on the input variable ut and the inner state of the preceding point in time st and the parameter vector v in accordance with the following rule:

$$s_{t+1} = NN(v, s_t, u_t), \tag{3}$$

where NN(.) designates a mapping rule which is defined by the neural network.

[0016] The arrangement which is known from Haykin and is designated as a Time Delay Recurrent Neural Network (TDRNN) is trained in a training phase in such a way that for an input variable ut, in each case a target variable y_t^d is determined in a real dynamic system. The tupel (input variable, determined target variable) is referred to as a training data item. A multiplicity of such training data items form a training data record.

[0017] The TDRNN is trained using the training data record. An overview of various training methods can also be found in Haykin.

[0018] It is to be noted at this point that only the output variable yt can be detected at a point in time t of the dynamic system 700. The inner system state st is not observable.

[0019] In the training phase, the following cost function E is usually minimized:

$$E = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2 \rightarrow \min_{f,g}, \tag{4}$$

where T designates a number of points in time to be taken into account.

[0020] From A. Zell, Simulation Neuronaler Netze, Addison-Wesley Publishing Company, Bonn, 1st Ed., 1994, pages 560-561, an arrangement of a plurality of interconnected neuron networks is known.

[0021] In the arrangement known from Zell, a plurality of hierarchically structured neural subnetworks are linked together in an overall structure in a parallel arrangement within the scope of what is referred to as a gating network.

[0022] In the gating network, an identical input vector is fed to each of the neural subnetworks. The neural subnetworks each determine an output vector in accordance with their internal structure. The output vectors of the neural subnetworks are summed in a linearly weighted fashion.

[0023] A training method for the gating network is also referred to in Zell.

[0024] The known arrangements and methods have, in particular, the disadvantage that identification or modeling of a dynamic system and determination of states of a dynamic system is possible only with insufficient precision.

SUMMARY OF THE INVENTION

[0025] The invention is therefore based on the problem of disclosing an arrangement with which a dynamic system can be modeled and a state of the dynamic system can be determined, and which permits the modeling and the determination with a greater degree of precision than in the known arrangements.

[0026] In addition, the invention is based on the problem of disclosing a method, a computer program event and a computer-readable storage medium with which a dynamic system can be modeled and a state of the dynamic system can be determined and which permit the modeling and the determination with a greater degree of precision than in the known arrangements.

[0027] A computer system for compensation of an inequilibrium state of a first technical system includes a first neural network which describes the first technical system and a second neural network which describes a second technical system. The first and the second neural networks are connected to one another in such a way that an inequilibrium state of the first technical system can be compensated by the second neural network.

[0028] In a method for the computer-supported compensation of an inequilibrium state of a first technical system, a first neural network, which describes the first technical system, is supplied with a first input variable. A first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network. The first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system. A second output variable, which describes a state of the second technical system, is determined for the second input variable using the second

neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0029] A computer program event which comprises a computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system: a first neural network, which describes the first technical system, is supplied with a first input variable; a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network; the first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system; and a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0030] A computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system: a first neural network, which describes the first technical system, is supplied with a first input variable; a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network; the first output variable is supplied, as a second input variable, to a second neural network which describes a second technical system; and a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0031] An inequilibrium state of a system is to be understood as a state of the system which according to predefinable criteria does not correspond to a selected state of the system, the equilibrium state.

[0032] The equilibrium state of the system can be distinguished, for example, by the fact that in this state the system has stability or effectiveness in terms of a transmission behavior of the system.

[0033] The invention has the particular advantage that a small amount of training data is necessary for training the arrangement in order to be able to carry out modeling of a dynamic system and the determination of a state of the dynamic system with sufficient precision using the trained arrangement.

[0034] Preferred developments of the invention emerge from the dependent claims.

[0035] The developments described below relate both to the method and the arrangement as well as to the computer program event and the computer-readable storage medium.

[0036] The invention and the developments described below can be implemented both by software and hardware, for example using a specific electric circuit.

[0037] The first neural network can be implemented in such a way that it has at least a first input computing element and a first output computing element.

[0038] The same applies to an implementation of the second neural network.

[0039] At least some of the computing elements are preferably artificial neurons.

[0040] In order to simplify training of one implementation of the invention, at least some of the connections between computing elements are of variable configuration.

[0041] In a further embodiment, at least some of the connections have identical weighting values.

[0042] For the sake of simplification during a description of a complex overall system, it is favorable to structure the complex overall system in such a way that the first technical system and the second technical system each describe a subsystem of the complex overall system.

[0043] However, in addition, the first technical system and the second technical system can also be identical.

[0044] Because the invention makes it possible to model a dynamic system with sufficient precision, one implementation is preferably used for determining a dynamic of a system.

[0045] In addition, one configuration is used for forecasting a future state of a system and for monitoring and/or controlling a system.

[0046] The system is preferably a chemical reactor.

[0047] Exemplary embodiments of the invention are illustrated in the figures and explained below in more detail

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] These and other objects and advantages of the present invention will become more apparent and more readily appreciated from the following description of the preferred embodiments, taken in conjunction with the accompanying drawings of which:

Figure 1 is a block diagram of a chemical reactor by which variables which are further processed by arrangement in accordance with a first exemplary embodiment are measured;

Figure 2 is a block diagram of an arrangement of neural networks in accordance with the first exemplary embodiment;

Figure 3 is a flowchart of a method in accordance with the first or second exemplary embodiment;

Figure 4 is a block diagram of a neural network in accordance with the first exemplary embodiment;

Figure 5 is a block diagram of a neural network in accordance with a second exemplary embodiment;

Figure 6 is a block diagram of a neural network during training in accordance with the second exemplary embodiment;

Figure 7 is a block diagram of a general description of a dynamic system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0049] Reference will now be made in detail to the preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to like elements throughout.

[0050] First exemplary embodiment: Chemical Reactor

[0051] Fig. 1 shows a chemical reactor 100 which is filled with a chemical substance 101 which is a mixture of a plurality of basic substances 103. The chemical reactor 100 comprises an agitator 102 with which the chemical substance 101 is agitated.

[0052] An injection device 150 injects the basic substances 103 into the reactor 100 separately from one another.

[0053] The basic substances 103 which are injected into the chemical reactor 100 react with one another during a predefinable time period in the chemical reactor 100, the chemical substance 101 being formed. A substance 104 which flows out of the reactor 100 is drawn off from the chemical reactor 100 via an output.

[0054] The injection device 150 is connected via a line to a control unit 105 with which monitored injection of any basic substance 103 into the reactor 100 can be set by a control signal 106.

[0055] In addition, a measuring device 107 is provided with which concentrations of the basic substances 103 contained in the chemical substance 101, a temperature in the reactor 100 and a pressure prevailing in the reactor 100 are measured.

[0056] Measurement signals 108 are fed to a computer 109, are digitized in the computer 109 by an input/output interface 110 and an analog/digital converter 111 and stored in a memory 112. A processor 113 is connected, as is the memory 112, to the analog/digital converter 111 via a bus 114. The computer 109 is also connected to the controller 105 of the injection device 150 via the input/output interface 110, and the computer 109 thus controls the injection of the basic substances 103 into the reactor 100.

[0057] The computer 109 is also connected via the input/output interface 110 to a keyboard 115, a computer mouse 116 and a display screen 117.

[0058] The chemical reactor 100 is subjected, as a dynamic technical system 200, to a dynamic process.

[0059] The chemical reactor 100 is described by a state description. The input variable ut is composed in this case of information relating to the temperature in the chemical reactor 100, to the pressure in the chemical reactor 100 and to the agitation frequency set at the point in time t. The input variable ut is therefore a three-dimensional vector.

[0060] The aim of the modeling of the chemical reactor 100 which is described below is to determine the dynamic profile of the concentrations of the basic substances 103 in the reactor

100 in order to make possible an efficient generation of a predefinable final product as a substance 104 which flows out.

[0061] An efficient generation of the predefinable final product is possible if the basic substances 103 are mixed in a ratio of the concentrations of the basic substances 103 which corresponds to the reaction to be carried out.

[0062] The dynamic profile of the concentrations of the basic substances 103 is determined using the arrangement which is described below and illustrated in Fig. 3.

[0063] For the sake of easier comprehension of the principles on which the arrangement is based, a basic structure 200 is illustrated in Fig. 2 as a two-part neural network in which a first neural network 201 and a second neural network 202 are connected in series.

[0064] The arrangements described below are each to be understood in such a way that each neuron layer or each sublayer has a predefinable number of neurons, i.e. computing elements.

[0065] In the basic structure illustrated in Fig. 2, the first neural network 201 and the second neural network 202 are linked to one another in such a way that outputs of the first neural network 201 are connected to inputs of the second neural network 202.

[0066] The first neural network 201 has a first input layer 210 with a predefinable number of input computing elements, i.e. input neurons, to which input variables ut can be supplied at a predefinable point in time t, i.e. in timing series values described below.

[0067] Furthermore, the first neural network 210 has output computing elements, i.e. output neurons, of a first output layer 220. The output neurons of the first output layer 220 are connected to the input neurons of the first input layer 210. The weightings of the connections are contained in a first connection matrix A.

[0068] The second neural network 202 is connected to the first neural network 201 in such a way that the output neurons of the first output layer 220 are connected to input neurons of a second input layer 230 in accordance with a structure given by a second connection matrix B.

[0069] In the second neural network 202, output neurons of a second output layer 240 are connected to the input neurons of the second input layer 230. Weightings of the connections are contained in a third connection matrix C.

[0070] The output variables yt for, in each case, one point in time t can be tapped at the output neurons of the second output layer 240.

[0071] The expanded arrangement illustrated in Fig. 4 is explained below on the basis of this basic structure.

[0072] Fig. 4 shows a third neural network 403 which is linked to the first neural network 401.

[0073] The third neural network 403 comprises a third input layer 450 with input neurons which are connected to neurons of a concealed layer 460 in accordance with the structure given by the first connection matrix A.

[0074] The third neural network 403 is connected to the first neural network 401 in such a way that the neurons of the concealed layer 460 are connected to the output neurons of the first output layer 420. Weightings of the connections are contained in a fourth connection matrix D.

[0075] The input neurons of the third input layer 450 are configured in such a way that the time series values ut can be supplied to them as input variables ut-1 at a predefinable point in time t-1.

[0076] The principle of what is referred to as shared weighting values (shared weights), i.e. the principle that equivalent connection matrices in a neural network have the same values at a particular point in time is implemented by the described configuration of the first neural network 401 and of the third neural network 403 in accordance with Fig. 4.

[0077] In the arrangement according to Fig. 4, in particular shared weighting values and the described configuration of the first input layer 410 and the third input layer 430 ensure that states st-1 and st, which are represented by the first output layer 420 and the concealed layer 460 describe two chronologically successive states t-1 and t of a system s.

[0078] The back propagation method is used as the training method. The training data record is acquired from the chemical reactor 400 in the following way.

[0079] The concentrations of the basic substances 103 are measured with the measuring device 407 with respect to predefined input variables and supplied to the computer 409, digitized there and grouped as time series values ut in chronological succession in a memory together with the corresponding input variables which correspond to the measured variables.

[0080] When the arrangement is trained, these time series values ut of the arrangement are supplied as a training data record together with the information relating to the predefined optimum ratio of the concentrations of the basic substances.

[0081] The arrangement from Fig. 4 is trained using the training data record.

[0082] In order to illustrate better the transformations achieved by the arrangement, steps of a method sequence 300 are illustrated in Fig. 3 with reference to the first exemplary embodiment.

[0083] In a first step 310, the arrangement is supplied with the time series values of the input variable ut which contains the information relating to the temperature, the pressure and the agitation frequency in the reactor 100.

[0084] In a second step 320, the input variable ut is used to determine an output variable ut of the first neural network, which output variable ut describes whether the basic substances are mixed together in the ratio which is optimized for an effective reaction of the basic substances and which constitutes what is referred to as an equilibrium state in the reactor. In this way, it is determined in the second step 320 whether a state in the reactor is in an equilibrium state or inequilibrium state.

[0085] In a third layer 330, the output variable st of the first neural network is supplied as an input variable to the second neural network.

[0086] In a fourth step 340, the second neural network determines the output variable yt which describes a dynamic change in the concentrations of the basic substances.

[0087] In the fourth step 340, a state with which the inequilibrium state can be compensated is determined in the reactor using the determined change in the concentrations of the basic substances.

[0088] The arrangement in Fig. 4 which is trained in accordance with the training method described above is used for the open-loop and closed-loop control of the injection process of the basic substances 103 into the chemical reactor 100.

[0089] The aim of the closed-loop and open-loop control is an automated continuous injection of the basic substances 103 into the chemical reactor 100 in such a way that the concentration ratio of the basic substances 103 in the reactor 100 has a constant ratio which is optimum for

the reaction to be carried out. An efficient generation of the predefinable end product as a substance 104 which flows out is thus possible.

[0090] For this purpose, a forecast value yt is determined in an application phase by the arrangement for a first input variable yt-1 at a point in time t-1 and for a second input variable ut at a point in time t, the forecast value ut being subsequently fed as control variable 420, after possible conditioning of the determined value, to the injection device 150 in order to control the injection of the basic substances 103 in the chemical reactor 100 (cf. Fig. 1).

[0091] The structuring of the arrangement in the first neural network and the second neural network, in particular an output layer which is formed by this structuring and which produces further fault signals, ensures that a small amount of training data is required to train the arrangement in order to ensure sufficient precision during modeling of the dynamic system.

2nd Exemplary embodiment: Exchange Rate Forecast

[0092] In a second exemplary embodiment, the arrangement described above according to Fig. 4 is used for an exchange rate forecast of a $/DM exchange rate.

[0093] A time series with time series values which each comprise information on economic figures, for example short-term and long-term interest rates in a $ area and a DM area, inflation rates and information relating to economic growth in the $ area and the DM area is fed to the arrangement as input variable ut.

[0094] A change in the $/DM exchange rate is forecast by the arrangement as output variable yt.

[0095] In the arrangement for the exchange rate forecast, in particular the connection matrices A, B, C and D have a particular configuration.

[0096] The first connection matrix A is filled with weightings in such a way that only a limited number of neurons of the first input layer of the first neural network (a maximum of seven neurons in this case) are assigned in each case to a neuron of the first output layer. Such a connection matrix is referred to as a "sparse connector".

[0097] In addition, the arrangement for the exchange rate forecast is configured in such a way that the first output layer of the first neural network has a large number of output neurons (200 output neurons in this case).

[0098] The second connection matrix B, in this case a highly dimensioned vector, is configured in such a way that the highly dimensional output variable st of the first neural network (dimension = 200) is mapped onto a one-dimensional variable using the connection matrix B. In particular, all the weightings of the second connection matrix B have the value 1.

[0099] Accordingly, the third connection matrix C only has a weighting value which additionally can assume only positive values.

[00100] The fourth connection matrix D is configured as a diagonal matrix in which all the diagonal values have the value 1.

[00101] The back propagation method is also used as the training method. A training data record is formed in the following way.

[00102] Known exchange rate changes are grouped as time series values ut in chronological succession together with the corresponding economic figures which correspond to the known exchange rate changes.

[00103] When the arrangement is trained, these time sequence values ut are supplied to the arrangement as a training data record.

[00104] During the training, a target function E, which is formed in the first output layer of the first neural network, has the following rule:

$$E = \frac{1}{T} \sum_{t} \ln\left(\frac{P_{t+1}}{P_t}\right) * z_t^a \rightarrow \max \qquad (5)$$

where:

t : index which describes a point in time

T : time interval under consideration

a : index for a neuron

ln(...) : natural logarithm

p_t, p_{t+1} : exchange rate at the point in time t or t+1

z_t^a : an exchanged quantity of money in DM which is assigned to a neuron a.

[00105] In addition, the following relationships which describe a dynamic exchange rate system apply:

$$m_{t+1}^a = m_t^a - z_t^a \quad \text{(market mechanism of the exchange rate system)} \qquad (6)$$

$$n_{t+1}^a = n_t^a - p_t * z_t^a \quad \text{(market mechanism of the exchange rate system)} \qquad (7)$$

$$\sum_a z_t^a = 0 \quad \text{(equilibrium condition of the exchange rate system)} \qquad (8)$$

where:

m_{t+1}^a, m_t^a : a quantity of money in \$, assigned to a neuron a, at a point in time t+1 or t

n_{t+1}^a, n_t^a : : a quantity of money in DM, assigned to a neuron a, at a point in time t+1 or t.

[00106] The arrangement for the exchange rate forecast is based on the following principles:

[00107] An inequilibrium state of the exchange rate system which is caused by an excess of an amount of money is compensated by a change in the exchange rate.

[00108] A state of the exchange rate system is described by a decision model which is implemented in the first neural network. In this case, in each case a neuron represents what is referred to as an "agent". Accordingly, the arrangement for the exchange rate forecast is also referred to as "multi-agent system".

[00109] In the second neural network, a market model is implemented with which an inequilibrium state which is generated by the decision model is compensated by a change in a state of an exchange rate market.

[00110] An inequilibrium state of the decision model is a cause of a change in state of the market model.

[00111] An alternative to the second exemplary embodiment is described below.

[00112] The alternative exemplary embodiment differs from the second exemplary embodiment in the following descriptive points. The corresponding arrangement of the alternative exemplary embodiment for the exchange rate forecast is illustrated in each case in Fig. 5 (application phase) and Fig. 6 (training phase).

[00113] Neuron connections which are broken are represented by dashed lines in the figures. Closed neuron connections are illustrated by continuous lines.

[00114] The original arrangement for the exchange rate forecast according to Fig. 4 can be changed to the effect that, instead of the third neural network, a fourth input layer 550 is used with input neurons which are connected to the output neurons of the second output layer 540 of the second neural network. Weightings of the connections are contained in a fifth connection matrix E. The connections are implemented in such a way that they are broken in an application phase and closed in a training phase.

[00115] In addition, the second output layer 540 of the second neural network has a feedback with which the output signals of the second output layer 540 are fed back into the second output layer 540. Weightings of the feedback are contained in a sixth connection matrix F. The feedback is implemented in such a way that it is closed in the application phase and interrupted in the training phase.

[00116] Moreover, the output neurons of the first output layer 520 of the first neural network are connected to the output neurons 540 of the second neural network. Weightings of the connections are contained in a seventh connection matrix G.

[00117] The connections between the second input layer 530 and the second output layer 540 are configured in such a way that they are closed in an application phase and interrupted in a training phase.

[00118] The fifth connection matrix E and the sixth connection matrix F is each an identity matrix Id.

[00119] The seventh connection matrix G is configured in such a way that a mapping which is carried out using the seventh connection matrix G obeys the following rule:

$$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}(\sum_a z_t^a) < 0 \qquad (9)$$

with:

$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}$: derived to $\ln\left(\frac{P_{t+1}}{P_t}\right)$.

[00120] Training of the alternative arrangement is carried out in accordance with the second exemplary embodiment. The alternative training arrangement is illustrated in Fig. 6.

[00121] During the training, the feedback is interrupted and the connections are broken between the second input layer 530 and the second output layer 540.

[00122] In the application phase, a change in an exchange rate with which an inequilibrium state of the market can be compensated can be tapped at the second output layer of the second neural network according to an iterative process.

[00123] Within the framework of the iterative process, the following fixed point problem is solved:

$$\ln\left(\frac{P_{t+1}}{P_t}\right)n+1 = \left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right)n + \varepsilon * \left(\sum_a a\left(\left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right)n\right)\right) \qquad (10)$$

$\ln(\frac{P_{t+1}}{P_t})n$ Fixed Point

where:

ε : weighting of the third connection matrix C, $\varepsilon > 0$.

N : index for an iteration step

[00124] The alternative arrangement for the exchange rate forecast is based on the following principles:

[00125] The exchange rate system is in equilibrium at every point in time t.

[00126] By changing a state of the market model, an inequilibrium state of the decision model can be prevented.

A possible implementation of the above-described second exemplary embodiment and a possible implementation of the alternative of the second exemplary embodiment are given below for the program SENN, version 2.3. The implementations each comprise three sections which each contain program code for processing in SENN, version 2.3.

[00127] The invention has been described in detail with particular reference to preferred embodiments thereof and examples, but it will be understood that variations and modifications can be effected within the spirit and scope of the invention.

ABSTRACT

SYSTEM FOR COMPENSATION OF AN INEQUILIBRIUM STATE

A first neural network describes a first technical system and a second neural network describes a second technical system. The first and the second neural networks are connected to one another in such a way that an inequilibrium state of the first technical system is compensated by the second neural network.

MARKED-UP COPY OF SUBSTITUTE SPECIFICATION

[Description] TITLE OF THE INVENTION

[ARRANGEMENT AND METHOD AND COMPUTER PROGRAM EVENT AND COMPUTER-READABLE STORAGE MEDIUM] SYSTEM FOR [THE COMPUTER-SUPPORTED] COMPENSATION OF AN INEQUILIBRIUM STATE [OF A TECHNICAL SYSTEM]

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is based on and hereby claims priority to German Application No. 199 28 776.7 filed on June 23, 1999, the contents of which are hereby incorporated by reference.

REFERENCE TO COMPUTER PROGRAM LISTING, COMPACT DISC APPENDIX

[0002] A compact disc is included herewith and incorporated by reference herein having thereon a computer program listing appendix in the ASCII uncompressed text format with ASCII carriage return, ASCII line feed and all control codes defined in ASCII, having computer compatibility with IBM PC/XT/AT or compatibles, having operating system compatibility with MS-Windows and including file PROGRA~1 (ProgramListing.txt in Windows) of 98,120 bytes, created on December 14, 2001.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0003] The invention relates to an [arrangement] system, a method, a computer program event and a computer-readable storage medium for the computer-supported compensation of an inequilibrium state of a technical system.

2. Description of the Related Art

[0004] From [1] S. Haykin, Neural Networks: a Comprehensive Foundation, McMillan College Publishing Company, 1994, pages 498-533, it is known to use a neural network to determine states of a dynamic system and a dynamic which is the basis for a dynamic system.

[0005] Generally, a dynamic process which occurs in a dynamic system is usually described by a state transition description which is not visible to an observer of the dynamic process, and a starting equation which describes observable variables of the technical dynamic process.

[0006] Such a structure is illustrated in Fig 7.

[0007] A dynamic system 700 is subject to the influence of an external input variable u with a predefinable dimension, an input variable ut at a point in time t being designated as ut:

$$u_t \in \Re^l,$$

l designating a natural number.

[0008] The input variable ut at a point in time t brings about a change in the dynamic process which occurs in the dynamic system 700.

[0009] An inner state st ($s_t \in \Re^m$) with a predefinable dimension m at a point in time t cannot be observed by an observer of the dynamic system 200.

[0010] A state transition of the inner state st of the dynamic process is brought about as a function of the inner state st and the input variable ut, and the state of the dynamic process changes into a subsequent state st+1 at a subsequent point in time t+1.

Here the following applies:

$$s_{t+1} = f(s_t, u_t). \tag{1}$$

where f(.) designates a general mapping rule.

[0011] An output variable y_t, which can be observed by an observer of the dynamic system 700, at a point in time t depends on the input variable u_t and on the inner state s_t.

[0012] The output variable y_t ($y_t \in \Re^n$) has a predefinable dimension n.

[0013] The dependence of the output variable y_t on the input variable u_t and the inner state s_t of the dynamic process is given by the following general rule:

$$y_t = g(s_t, u_t), \tag{2}$$

where g(.) designates a general mapping rule.

[0014] In [1] Haykin, an arrangement of interconnected computing elements in the form of a neural network of interconnected neurons is used to describe the dynamic system 700. The connections between the neurons of the neural network are weighted. The weightings of the neural network are combined in a parameter vector v.

[0015] An inner state of a dynamic system, which is subject to a dynamic process, thus depends on the input variable ut and the inner state of the preceding point in time st and the parameter vector v in accordance with the following rule:

$$s_{t+1} = NN(v, s_t, u_t), \tag{3}$$

where NN(.) designates a mapping rule which is defined by the neural network.

[0016] The arrangement which is known from [1] Haykin and is designated as a Time Delay Recurrent Neural Network (TDRNN) is trained in a training phase in such a way that for an input variable ut, in each case a target variable y_t^d is determined in a real dynamic system. The tupel (input variable, determined target variable) is referred to as a training data item. A multiplicity of such training data items form a training data record.

[0017] The TDRNN is trained using the training data record. An overview of various training methods can also be found in [1] Haykin.

[0018] It is to be noted at this point that only the output variable yt can be detected at a point in time t of the dynamic system 700. The inner system state st is not observable.

[0019] In the training phase, the following cost function E is usually minimized:

$$E = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2 \rightarrow \min_{f,g}, \tag{4}$$

where T designates a number of points in time to be taken into account.

[0020] From [2] A. Zell, Simulation Neuronaler Netze, Addison-Wesley Publishing Company, Bonn, 1st Ed., 1994, pages 560-561, an arrangement of a plurality of interconnected neuron networks is known.

[0021] In the arrangement known from [2] Zell, a plurality of hierarchically structured neural subnetworks are linked together in an overall structure in a parallel arrangement within the scope of what is referred to as a gating network.

[0022] In the gating network, an identical input vector is fed to each of the neural subnetworks. The neural subnetworks each determine an output vector in accordance with their internal structure. The output vectors of the neural subnetworks are summed in a linearly weighted fashion.

[0023] A training method for the gating network is also referred to in [2] Zell.

[0024] The known arrangements and methods have, in particular, the disadvantage that identification or modeling of a dynamic system and determination of states of a dynamic system is possible only with insufficient precision.

SUMMARY OF THE INVENTION

[0025] The invention is therefore based on the problem of disclosing an arrangement with which a dynamic system can be modeled and a state of the dynamic system can be determined, and which [arrangement] permits the modeling and the determination with a greater degree of precision than in the known arrangements.

[0026] In addition, the invention is based on the problem of disclosing a method, a computer program event and a computer-readable storage medium with which a dynamic system can be modeled and a state of the dynamic system can be determined and which permit the modeling and the determination with a greater degree of precision than in the known arrangements. [The problems are achieved by means of the arrangement and the methods having the features according to the independent claims.]

[0027] [An arrangement] A computer system for [the computer-supported] compensation of an inequilibrium state of a first technical system [comprises] includes a first neural network which describes the first technical system and a second neural network which describes a second technical system. The first and the second neural [network] networks are connected to one another in such a way that an inequilibrium state of the first technical system can be compensated by the second neural network.

[0028] In a method for the computer-supported compensation of an inequilibrium state of a first technical system, a first neural network, which describes the first technical system, is supplied with a first input variable. A first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network. The first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system. A second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0029] A computer program event which comprises a computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system: [-] a first neural network, which describes the first technical system, is supplied with a first input variable; [-] a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network; [-] the first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system; [-] <u>and</u> a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0030] A computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system: [-] a first neural network, which describes the first technical system, is supplied with a first input variable; [-] a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network; [-] the first output variable is supplied, as a second input variable, to a second neural network which describes a second technical system; [-] <u>and</u> a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

[0031] An inequilibrium state of a system is to be understood as a state of the system which according to predefinable criteria does not correspond to a selected state of the system, the equilibrium state.

[0032] The equilibrium state of the system can be distinguished, for example, by the fact that in this state the system has stability or effectiveness in terms of a transmission behavior of the system.

[0033] The invention has the particular advantage that a small amount of training data is necessary for training the arrangement in order to be able to carry out modeling of a dynamic system and the determination of a state of the dynamic system with sufficient precision using the trained arrangement.

[0034] Preferred developments of the invention emerge from the dependent claims.

[0035] The developments described below relate both to the method and the arrangement as well as to the computer program event and the computer-readable storage medium.

[0036] The invention and the developments described below can be implemented both by software and hardware, for example using a specific electric circuit.

[0037] The first neural network can be implemented in such a way that it has at least a first input computing element and a first output computing element.

[0038] The same applies to an implementation of the second neural network.

[0039] At least some of the computing elements are preferably artificial neurons.

[0040] In order to simplify training of one implementation of the invention, at least some of the connections between computing elements are of variable configuration.

[0041] In a further embodiment, at least some of the connections have identical weighting values.

[0042] For the sake of simplification during a description of a complex overall system, it is favorable to structure the complex overall system in such a way that the first technical system and the second technical system each describe a subsystem of the complex overall system.

[0043] However, in addition, the first technical system and the second technical system can also be identical.

[0044] Because the invention makes it possible to model a dynamic system with sufficient precision, one implementation is preferably used for determining a dynamic of a system.

[0045] In addition, one configuration is used for forecasting a future state of a system and for monitoring and/or controlling a system.

[0046] The system is preferably a chemical reactor.

[0047] Exemplary embodiments of the invention are illustrated in the figures and explained below in more detail[. In said figures:]

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] [****] These and other objects and advantages of the present invention will become more apparent and more readily appreciated from the following description of the preferred embodiments, taken in conjunction with the accompanying drawings of which:

Figure 1 [shows an outline] is a block diagram of a chemical reactor by which variables which are further processed [with the] by arrangement in accordance with a first exemplary embodiment are measured;

Figure 2 [shows an outline] is a block diagram of an arrangement of neural networks in accordance with the first exemplary embodiment;

Figure 3 [shows an outline which describes] is a flowchart of a method [sequence] in accordance with the first or second exemplary embodiment;

Figure 4 [shows an outline of an arrangement] is a block diagram of a neural network in accordance with the first exemplary embodiment;

Figure 5 [shows an outline of an arrangement] is a block diagram of a neural network in accordance with a second exemplary embodiment;

Figure 6 [shows an outline of an arrangement] is a block diagram of a neural network during training in accordance with the second exemplary embodiment;

Figure 7 [shows an outline] is a block diagram of a general description of a dynamic system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0049] Reference will now be made in detail to the preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to like elements throughout.

[0050] First exemplary embodiment: Chemical Reactor

[0051] Fig. 1 shows a chemical reactor 100 which is filled with a chemical substance 101 which is a mixture of a plurality of basic substances 103. The chemical reactor 100 comprises an agitator 102 with which the chemical substance 101 is agitated.

[0052] An injection device 150 injects the basic substances 103 into the reactor 100 separately from one another.

[0053] The basic substances 103 which are injected into the chemical reactor 100 react with one another during a predefinable time period in the chemical reactor 100, the chemical substance 101 being formed. A substance 104 which flows out of the reactor 100 is drawn off from the chemical reactor 100 via an output.

[0054] The injection device 150 is connected via a line to a control unit 105 with which monitored injection of any basic substance 103 into the reactor 100 can be set by [means of] a control signal 106.

[0055] In addition, a measuring device 107 is provided with which concentrations of the basic substances 103 contained in the chemical substance 101, a temperature in the reactor 100 and a pressure prevailing in the reactor 100 are measured.

[0056] Measurement signals 108 are fed to a computer 109, are digitized in the computer 109 by [means of] an input/output interface 110 and an analog/digital converter 111 and stored in a memory 112. A processor 113 is connected, as is the memory 112, to the analog/digital converter 111 via a bus 114. The computer 109 is also connected to the controller 105 of the injection device 150 via the input/output interface 110, and the computer 109 thus controls the injection of the basic substances 103 into the reactor 100.

[0057] The computer 109 is also connected via the input/output interface 110 to a keyboard 115, a computer mouse 116 and a display screen 117.

[0058] The chemical reactor 100 is subjected, as a dynamic technical system 200, to a dynamic process.

[0059] The chemical reactor 100 is described by [means of] a state description. The input variable ut is composed in this case of information relating to the temperature in the chemical reactor 100, to the pressure in the chemical reactor 100 and to the agitation frequency set at the point in time t. The input variable ut is therefore a three-dimensional vector.

[0060] The aim of the modeling of the chemical reactor 100 which is described below is to determine the dynamic profile of the concentrations of the basic substances 103 in the reactor 100 in order to make possible an efficient generation of a predefinable final product as a substance 104 which flows out.

[0061] An efficient generation of the predefinable final product is possible if the basic substances 103 are mixed in a ratio of the concentrations of the basic substances 103 which corresponds to the reaction to be carried out.

[0062] The dynamic profile of the concentrations of the basic substances 103 is determined using the arrangement which is described below and illustrated in Fig. 3.

[0063] For the sake of easier comprehension of the principles on which the arrangement is based, a basic structure 200 is illustrated in Fig. 2 as a two-part neural network in which a first neural network 201 and a second neural network 202 are connected in series.

[0064] The arrangements described below are each to be understood in such a way that each neuron layer or each sublayer has a predefinable number of neurons, i.e. computing elements.

[0065] In the basic structure illustrated in Fig. 2, the first neural network 201 and the second neural network 202 are linked to one another in such a way that outputs of the first neural network 201 are connected to inputs of the second neural network 202.

[0066] The first neural network 201 has a first input layer 210 with a predefinable number of input computing elements, i.e. input neurons, to which input variables ut can be supplied at a predefinable point in time t, i.e. in timing series values described below.

[0067] Furthermore, the first neural network 210 has output computing elements, i.e. output neurons, of a first output layer 220. The output neurons of the first output layer 220 are

connected to the input neurons of the first input layer 210. The weightings of the connections are contained in a first connection matrix A.

[0068] The second neural network 202 is connected to the first neural network 201 in such a way that the output neurons of the first output layer 220 are connected to input neurons of a second input layer 230 in accordance with a structure given by a second connection matrix B.

[0069] In the second neural network 202, output neurons of a second output layer 240 are connected to the input neurons of the second input layer 230. Weightings of the connections are contained in a third connection matrix C.

[0070] The output variables yt for, in each case, one point in time t can be tapped at the output neurons of the second output layer 240.

[0071] The expanded arrangement illustrated in Fig. 4 is explained below on the basis of this basic structure.

[0072] Fig. 4 shows a third neural network 403 which is linked to the first neural network 401.

[0073] The third neural network 403 comprises a third input layer 450 with input neurons which are connected to neurons of a concealed layer 460 in accordance with the structure given by the first connection matrix A.

[0074] The third neural network 403 is connected to the first neural network 401 in such a way that the neurons of the concealed layer 460 are connected to the output neurons of the first output layer 420. Weightings of the connections are contained in a fourth connection matrix D.

[0075] The input neurons of the third input layer 450 are configured in such a way that the time series values ut can be supplied to them as input variables ut-1 at a predefinable point in time t-1.

[0076] The principle of what is referred to as shared weighting values (shared weights), i.e. the principle that equivalent connection matrices in a neural network have the same values at a particular point in time is implemented by [means of] the described configuration of the first neural network 401 and of the third neural network 403 in accordance with Fig. 4.

[0077] In the arrangement according to Fig. 4, in particular shared weighting values and the described configuration of the first input layer 410 and the third input layer 430 ensure that

states st-1 and st, which are represented by the first output layer 420 and the concealed layer 460 describe two chronologically successive states t-1 and t of a system s.

[0078] The back propagation method is used as the training method. The training data record is acquired from the chemical reactor 400 in the following way.

[0079] The concentrations of the basic substances 103 are measured with the measuring device 407 with respect to predefined input variables and supplied to the computer 409, digitized there and grouped as time series values ut in chronological succession in a memory together with the corresponding input variables which correspond to the measured variables.

[0080] When the arrangement is trained, these time series values ut of the arrangement are supplied as a training data record together with the information relating to the predefined optimum ratio of the concentrations of the basic substances.

[0081] The arrangement from Fig. 4 is trained using the training data record.

[0082] In order to illustrate better the transformations achieved by [means of] the arrangement, steps of a method sequence 300 are illustrated in Fig. 3 with reference to the first exemplary embodiment.

[0083] In a first step 310, the arrangement is supplied with the time series values of the input variable ut which contains the information relating to the temperature, the pressure and the agitation frequency in the reactor 100.

[0084] In a second step 320, the input variable ut is used to determine an output variable ut of the first neural network, which output variable ut describes whether the basic substances are mixed together in the ratio which is optimized for an effective reaction of the basic substances and which constitutes what is referred to as an equilibrium state in the reactor. In this way, it is determined in the second step 320 whether a state in the reactor is in an equilibrium state or inequilibrium state.

[0085] In a third layer 330, the output variable st of the first neural network is supplied as an input variable to the second neural network.

[0086] In a fourth step 340, the second neural network determines the output variable yt which describes a dynamic change in the concentrations of the basic substances.

[0087] In the fourth step 340, a state with which the inequilibrium state can be compensated is determined in the reactor using the determined change in the concentrations of the basic substances.

[0088] The arrangement in Fig. 4 which is trained in accordance with the training method described above is used for the open-loop and closed-loop control of the injection process of the basic substances 103 into the chemical reactor 100.

[0089] The aim of the closed-loop and open-loop control is an automated continuous injection of the basic substances 103 into the chemical reactor 100 in such a way that the concentration ratio of the basic substances 103 in the reactor 100 has a constant ratio which is optimum for the reaction to be carried out. An efficient generation of the predefinable end product as a substance 104 which flows out is thus possible.

[0090] For this purpose, a forecast value yt is determined in an application phase by the arrangement for a first input variable yt-1 at a point in time t-1 and for a second input variable ut at a point in time t, [said] the forecast value ut being subsequently fed as control variable 420, after possible conditioning of the determined value, to the injection device 150 in order to control the injection of the basic substances 103 in the chemical reactor 100 (cf. Fig. 1).

[0091] The structuring of the arrangement in the first neural network and the second neural network, in particular an output layer which is formed by this structuring and which produces further fault signals, ensures that a small amount of training data is required to train the arrangement in order to ensure sufficient precision during modeling of the dynamic system.

2nd Exemplary embodiment: Exchange Rate Forecast

[0092] In a second exemplary embodiment, the arrangement described above according to Fig. 4 is used for an exchange rate forecast of a $/DM exchange rate.

[0093] A time series with time series values which each comprise information on economic figures, for example short-term and long-term interest rates in a $ area and a DM area, inflation rates and information relating to economic growth in the $ area and the DM area is fed to the arrangement as input variable ut.

[0094] A change in the $/DM exchange rate is forecast by the arrangement as output variable yt.

[0095] In the arrangement for the exchange rate forecast, in particular the connection matrices A, B, C and D have a particular configuration.

[0096] The first connection matrix A is filled with weightings in such a way that only a limited number of neurons of the first input layer of the first neural network (a maximum of seven neurons in this case) are assigned in each case to a neuron of the first output layer. Such a connection matrix is referred to as a "sparse connector".

[0097] In addition, the arrangement for the exchange rate forecast is configured in such a way that the first output layer of the first neural network has a large number of output neurons (200 output neurons in this case).

[0098] The second connection matrix B, in this case a highly dimensioned vector, is configured in such a way that the highly dimensional output variable st of the first neural network (dimension = 200) is mapped onto a one-dimensional variable using the connection matrix B. In particular, all the weightings of the second connection matrix B have the value 1.

[0099] Accordingly, the third connection matrix C only has a weighting value which additionally can assume only positive values.

[00100] The fourth connection matrix D is configured as a diagonal matrix in which all the diagonal values have the value 1.

[00101] The back propagation method is also used as the training method. A training data record is formed in the following way.

[00102] Known exchange rate changes are grouped as time series values ut in chronological succession together with the corresponding economic figures which correspond to the known exchange rate changes.

[00103] When the arrangement is trained, these time sequence values ut are supplied to the arrangement as a training data record.

[00104] During the training, a target function E, which is formed in the first output layer of the first neural network, has the following rule:

$$E = \frac{1}{T} \sum_{t} \ln \left(\frac{p_{t+1}}{p_t} \right) * z_t^a \rightarrow \max \quad (5)$$

where:

t : index which describes a point in time

T : time interval under consideration

a : index for a neuron

ln(...) : natural logarithm

p_t, p_{t+1} : exchange rate at the point in time t or t+1

z_t^a : an exchanged quantity of money in DM which is assigned to a neuron a.

[00105] In addition, the following relationships which describe a dynamic exchange rate system apply:

$$m_{t+1}^a = m_t^a - z_t^a$$ (market mechanism of the exchange rate system) (6)

$$n_{t+1}^a = n_t^a - p_t * z_t^a$$ (market mechanism of the exchange rate system) (7)

$$\sum_{a} z_t^a = 0$$ (equilibrium condition of the exchange rate system) (8)

where:

m_{t+1}^a, m_t^a : a quantity of money in $, assigned to a neuron a, at a point in time t+1 or t

n_{t+1}^a, n_t^a : : a quantity of money in DM, assigned to a neuron a, at a point in time t+1 or t.

[00106] The arrangement for the exchange rate forecast is based on the following principles:

[00107] An inequilibrium state of the exchange rate system which is caused by an excess of an amount of money is compensated by a change in the exchange rate.

[00108] A state of the exchange rate system is described by a decision model which is implemented in the first neural network. In this case, in each case a neuron represents what is referred to as an "agent". Accordingly, the arrangement for the exchange rate forecast is also referred to as "multi-agent system".

[00109] In the second neural network, a market model is implemented with which an inequilibrium state which is generated by the decision model is compensated by a change in a state of an exchange rate market.

[00110] An inequilibrium state of the decision model is a cause of a change in state of the market model.

[00111] An alternative to the second exemplary embodiment is described below.

[00112] The alternative exemplary embodiment differs from the second exemplary embodiment in the following descriptive points. The corresponding arrangement of the alternative exemplary embodiment for the exchange rate forecast is illustrated in each case in Fig. 5 (application phase) and Fig. 6 (training phase).

[00113] Neuron connections which are broken are represented by dashed lines in the figures. Closed neuron connections are illustrated by continuous lines.

[00114] The original arrangement for the exchange rate forecast according to Fig. 4 can be changed to the effect that, instead of the third neural network, a fourth input layer 550 is used with input neurons which are connected to the output neurons of the second output layer 540 of the second neural network. Weightings of the connections are contained in a fifth connection matrix E. The connections are implemented in such a way that they are broken in an application phase and closed in a training phase.

[00115] In addition, the second output layer 540 of the second neural network has a feedback with which the output signals of the second output layer 540 are fed back into the second output layer 540. Weightings of the feedback are contained in a sixth connection matrix F. The feedback is implemented in such a way that it is closed in the application phase and interrupted in the training phase.

[00116] Moreover, the output neurons of the first output layer 520 of the first neural network are connected to the output neurons 540 of the second neural network. Weightings of the connections are contained in a seventh connection matrix G.

[00117] The connections between the second input layer 530 and the second output layer 540 are configured in such a way that they are closed in an application phase and interrupted in a training phase.

[00118] The fifth connection matrix E and the sixth connection matrix F is each an identity matrix Id.

[00119] The seventh connection matrix G is configured in such a way that a mapping which is carried out using the seventh connection matrix G obeys the following rule:

$$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}(\sum_a z_t^a) < 0 \qquad (9)$$

with:

$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}$: derived to $\ln\left(\frac{p_{t+1}}{p_t}\right)$.

[00120] Training of the alternative arrangement is carried out in accordance with the second exemplary embodiment. The alternative training arrangement is illustrated in Fig. 6.

[00121] During the training, the feedback is interrupted and the connections are broken between the second input layer 530 and the second output layer 540.

[00122] In the application phase, a change in an exchange rate with which an inequilibrium state of the market can be compensated can be tapped at the second output layer of the second neural network according to an iterative process.

[00123] Within the framework of the iterative process, the following fixed point problem is solved:

$$\ln\left(\frac{P_{t+1}}{P_t}\right) n + 1 = \left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right) n + \varepsilon * \left(\sum_{a} a\left(\left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right) n\right)\right) \qquad (10)$$

$\ln\left(\frac{P_{t+1}}{P_t}\right) n$ Fixed Point

where:

ε : weighting of the third connection matrix C, $\varepsilon > 0$.

N : index for an iteration step

[00124] The alternative arrangement for the exchange rate forecast is based on the following principles:

[00125] The exchange rate system is in equilibrium at every point in time t.

[00126] By changing a state of the market model, an inequilibrium state of the decision model can be prevented.

A possible implementation of the above-described second exemplary embodiment and a possible implementation of the alternative of the second exemplary embodiment are given below for the program SENN, version 2.3. The implementations each comprise three sections which each contain [a] program code[, said codes being necessary] for processing in SENN, version 2.3.

[00127] <u>The invention has been described in detail with particular reference to preferred embodiments thereof and examples, but it will be understood that variations and modifications can be effected within the spirit and scope of the invention.</u>

[Possible implementation of the second exemplary embodiment:]

[Program listing on pages 22-67 has been deleted and stored on the CD-R disc submitted herewith]

[In this document the following references are cited.]

[1] [S. Haykin, Neural Networks: A Comprehensive Foundation, Mc Millan College Publishing Company, ISBN 0-02-352761-7, S. 498-533, 1994.]

[2] [A. Zell, Simulation Neuronaler Netze, Addison-Wesley Publishing Company, S.560-561, 1. Auflage, Bonn, 1994]

10/019005
JC13 Rec'd PCT/PTO 26 DEC 2001

5/prts

Description

Arrangement and method and computer program event and computer-readable storage medium for the computer-supported compensation of an inequilibrium state of a technical system

The invention relates to an arrangement, a method, a computer program event and a computer-readable storage medium for the computer-supported compensation of an inequilibrium state of a technical system.

From [1] it is known to use a neural network to determine states of a dynamic system and a dynamic which is the basis for a dynamic system.

Generally, a dynamic process which occurs in a dynamic system is usually described by a state transition description which is not visible to an observer of the dynamic process, and a starting equation which describes observable variables of the technical dynamic process.

Such a structure is illustrated in Fig 7.

A dynamic system 700 is subject to the influence of an external input variable u with a predefinable dimension, an input variable u_t at a point in time t being designated as u_t:

$$u_t \in \Re^1,$$

l designating a natural number.

The input variable u_t at a point in time t brings about a change in the dynamic process which occurs in the dynamic system 700.

An inner state s_t ($s_t \in \Re^m$) with a predefinable dimension m at a point in time t cannot be observed by an observer of the dynamic system 200.

A state transition of the inner state s_t of the dynamic process is brought about as a function of the inner state s_t and the input variable u_t, and the state of the dynamic process changes into a subsequent state s_{t+1} at a subsequent point in time t+1.

Here the following applies:

$$s_{t+1} = f(s_t, u_t). \tag{1}$$

where f(.) designates a general mapping rule.

An output variable y_t, which can be observed by an observer of the dynamic system 700, at a point in time t depends on the input variable u_t and on the inner state s_t.

The output variable y_t ($y_t \in \Re^n$) has a predefinable dimension n.

The dependence of the output variable y_t on the input variable u_t and the inner state s_t of the dynamic process is given by the following general rule:

$$y_t = g(s_t, u_t), \tag{2}$$

where g(.) designates a general mapping rule.

In [1], an arrangement of interconnected computing elements in the form of a neural network of interconnected neurons is used to describe the dynamic system 700. The connections between the neurons of the neural network are weighted. The

weightings of the neural network are combined in a parameter vector v.

An inner state of a dynamic system, which is subject to a dynamic process, thus depends on the input variable u_t and the inner state of the preceding point in time s_t and the parameter vector v in accordance with the following rule:

$$s_{t+1} = NN(v, s_t, u_t), \quad (3)$$

where NN(.) designates a mapping rule which is defined by the neural network.

The arrangement which is known from [1] and is designated as a Time Delay Recurrent Neural Network (TDRNN) is trained in a training phase in such a way that for an input variable u_t, in each case a target variable y_t^d is determined in a real dynamic system. The tupel (input variable, determined target variable) is referred to as a training data item. A multiplicity of such training data items form a training data record.

The TDRNN is trained using the training data record. An overview of various training methods can also be found in [1].

It is to be noted at this point that only the output variable y_t can be detected at a point in time t of the dynamic system 700. The inner system state s_t is not observable.

In the training phase, the following cost function E is usually minimized:

$$E = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2 \rightarrow \min_{f,g}, \quad (4)$$

where T designates a number of points in time to be taken into account.

From [2], an arrangement of a plurality of interconnected neuron networks is known.

In the arrangement known from [2], a plurality of hierarchically structured neural subnetworks are linked together in an overall structure in a parallel arrangement within the scope of what is referred to as a gating network.

In the gating network, an identical input vector is fed to each of the neural subnetworks. The neural subnetworks each determine an output vector in accordance with their internal structure. The output vectors of the neural subnetworks are summed in a linearly weighted fashion.

A training method for the gating network is also referred to in [2].

The known arrangements and methods have, in particular, the disadvantage that identification or modeling of a dynamic system and determination of states of a dynamic system is possible only with insufficient precision.

The invention is therefore based on the problem of disclosing an arrangement with which a dynamic system can be modeled and a state of the dynamic system can be determined, and which arrangement permits the modeling and the determination with a greater degree of precision than in the known arrangements.

In addition, the invention is based on the problem of disclosing a method, a computer program event and a computer-readable storage medium with which a dynamic system can be modeled and a state of the dynamic system can be determined and which permit the modeling and the determination with a greater degree of precision than in the known arrangements.

The problems are achieved by means of the arrangement and the methods having the features according to the independent claims.

An arrangement for the computer-supported compensation of an inequilibrium state of a first technical system comprises a first neural network which describes the first technical system and a second neural network which describes a second technical system. The first and the second neural network are connected to one another in such a way that an inequilibrium state of the first technical system can be compensated by the second neural network.

In a method for the computer-supported compensation of an inequilibrium state of a first technical system, a first neural network, which describes the first technical system, is supplied with a first input variable. A first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network. The first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system. A second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

A computer program event which comprises a computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system:

- a first neural network, which describes the first technical system, is supplied with a first input variable;

- a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network;
- the first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system;
- a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

A computer-readable storage medium on which a program is stored makes it possible, after it has been loaded into a memory of a computer, for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system:
- a first neural network, which describes the first technical system, is supplied with a first input variable;
- a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network;
- the first output variable is supplied, as a second input variable, to a second neural network which describes a second technical system;
- a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

An inequilibrium state of a system is to be understood as a state of the system which according to predefinable criteria does not correspond to a selected state of the system, the equilibrium state.

The equilibrium state of the system can be distinguished, for example, by the fact that in this state the system has stability or effectiveness in terms of a transmission behavior of the system.

The invention has the particular advantage that a small amount of training data is necessary for training the arrangement in order to be able to carry out modeling of a dynamic system and the determination of a state of the dynamic system with sufficient precision using the trained arrangement.

Preferred developments of the invention emerge from the dependent claims.

The developments described below relate both to the method and the arrangement as well as to the computer program event and the computer-readable storage medium.

The invention and the developments described below can be implemented both by software and hardware, for example using a specific electric circuit.

The first neural network can be implemented in such a way that it has at least a first input computing element and a first output computing element.

The same applies to an implementation of the second neural network.

At least some of the computing elements are preferably artificial neurons.

In order to simplify training of one implementation of the invention, at least some of the connections between computing elements are of variable configuration.

In a further embodiment, at least some of the connections have identical weighting values.

For the sake of simplification during a description of a complex overall system, it is favorable to structure the complex overall system in such a way that the first technical system and the second technical system each describe a subsystem of the complex overall system.

However, in addition, the first technical system and the second technical system can also be identical.

Because the invention makes it possible to model a dynamic system with sufficient precision, one implementation is preferably used for determining a dynamic of a system.

In addition, one configuration is used for forecasting a future state of a system and for monitoring and/or controlling a system.

The system is preferably a chemical reactor.

Exemplary embodiments of the invention are illustrated in the figures and explained below in more detail. In said figures:

Figure 1 shows an outline of a chemical reactor by which variables which are further processed with the arrangement in accordance with a first exemplary embodiment are measured;

Figure 2 shows an outline of an arrangement in accordance with the first exemplary embodiment;

Figure 3 shows an outline which describes a method sequence in accordance with the first or second exemplary embodiment;

Figure 4 shows an outline of an arrangement in accordance with the first exemplary embodiment;

Figure 5 shows an outline of an arrangement in accordance with a second exemplary embodiment;

Figure 6 shows an outline of an arrangement during training in accordance with the second exemplary embodiment;

Figure 7 shows an outline of a general description of a dynamic system.

First exemplary embodiment: Chemical Reactor

Fig.1 shows a chemical reactor 100 which is filled with a chemical substance 101 which is a mixture of a plurality of basic substances 103. The chemical reactor 100 comprises an agitator 102 with which the chemical substance 101 is agitated.

An injection device 150 injects the basic substances 103 into the reactor 100 separately from one another.

The basic substances 103 which are injected into the chemical reactor 100 react with one another during a predefinable time period in the chemical reactor 100, the chemical substance 101 being formed. A substance 104 which flows out of the reactor 100 is drawn off from the chemical reactor 100 via an output.

The injection device 150 is connected via a line to a control unit 105 with which monitored injection of any basic substance 103 into the reactor 100 can be set by means of a control signal 106.

In addition, a measuring device 107 is provided with which concentrations of the basic substances 103 contained in the

chemical substance 101, a temperature in the reactor 100 and a pressure prevailing in the reactor 100 are measured.

Measurement signals 108 are fed to a computer 109, are digitized in the computer 109 by means of an input/output interface 110 and an analog/digital converter 111 and stored in a memory 112. A processor 113 is connected, as is the memory 112, to the analog/digital converter 111 via a bus 114. The computer 109 is also connected to the controller 105 of the injection device 150 via the input/output interface 110, and the computer 109 thus controls the injection of the basic substances 103 into the reactor 100.

The computer 109 is also connected via the input/output interface 110 to a keyboard 115, a computer mouse 116 and a display screen 117.

The chemical reactor 100 is subjected, as a dynamic technical system 200, to a dynamic process.

The chemical reactor 100 is described by means of a state description. The input variable u_t is composed in this case of information relating to the temperature in the chemical reactor 100, to the pressure in the chemical reactor 100 and to the agitation frequency set at the point in time t. The input variable u_t is therefore a three-dimensional vector.

The aim of the modeling of the chemical reactor 100 which is described below is to determine the dynamic profile of the concentrations of the basic substances 103 in the reactor 100 in order to make possible an efficient generation of a predefinable final product as a substance 104 which flows out.

An efficient generation of the predefinable final product is possible if the basic substances 103 are mixed in a ratio of

the concentrations of the basic substances 103 which corresponds to the reaction to be carried out.

The dynamic profile of the concentrations of the basic substances 103 is determined using the arrangement which is described below and illustrated in Fig. 3.

For the sake of easier comprehension of the principles on which the arrangement is based, a basic structure 200 is illustrated in Fig. 2 as a two-part neural network in which a first neural network 201 and a second neural network 202 are connected in series.

The arrangements described below are each to be understood in such a way that each neuron layer or each sublayer has a predefinable number of neurons, i.e. computing elements.

In the basic structure illustrated in Fig. 2, the first neural network 201 and the second neural network 202 are linked to one another in such a way that outputs of the first neural network 201 are connected to inputs of the second neural network 202.

The first neural network 201 has a first input layer 210 with a predefinable number of input computing elements, i.e. input neurons, to which input variables u_t can be supplied at a predefinable point in time t, i.e. in timing series values described below.

Furthermore, the first neural network 210 has output computing elements, i.e. output neurons, of a first output layer 220. The output neurons of the first output layer 220 are connected to the input neurons of the first input layer 210. The weightings of the connections are contained in a first connection matrix A.

The second neural network 202 is connected to the first neural network 201 in such a way that the output neurons of the first output layer 220 are connected to input neurons of a second input layer 230 in accordance with a structure given by a second connection matrix B.

In the second neural network 202, output neurons of a second output layer 240 are connected to the input neurons of the second input layer 230. Weightings of the connections are contained in a third connection matrix C.

The output variables y_t for, in each case, one point in time t can be tapped at the output neurons of the second output layer 240.

The expanded arrangement illustrated in Fig. 4 is explained below on the basis of this basic structure.

Fig.4 shows a third neural network 403 which is linked to the first neural network 401.

The third neural network 403 comprises a third input layer 450 with input neurons which are connected to neurons of a concealed layer 460 in accordance with the structure given by the first connection matrix A.

The third neural network 403 is connected to the first neural network 401 in such a way that the neurons of the concealed layer 460 are connected to the output neurons of the first output layer 420. Weightings of the connections are contained in a fourth connection matrix D.

The input neurons of the third input layer 450 are configured in such a way that the time series values u_t can be supplied to them as input variables u_{t-1} at a predefinable point in time t-1.

The principle of what is referred to as shared weighting values (shared weights), i.e. the principle that equivalent connection matrices in a neural network have the same values at a particular point in time is implemented by means of the described configuration of the first neural network 401 and of the third neural network 403 in accordance with Fig. 4.

In the arrangement according to Fig. 4, in particular shared weighting values and the described configuration of the first input layer 410 and the third input layer 430 ensure that states s_{t-1} and s_t, which are represented by the first output layer 420 and the concealed layer 460 describe two chronologically successive states t-1 and t of a system s.

The back propagation method is used as the training method. The training data record is acquired from the chemical reactor 400 in the following way.

The concentrations of the basic substances 103 are measured with the measuring device 407 with respect to predefined input variables and supplied to the computer 409, digitized there and grouped as time series values u_t in chronological succession in a memory together with the corresponding input variables which correspond to the measured variables.

When the arrangement is trained, these time series values u_t of the arrangement are supplied as a training data record together with the information relating to the predefined optimum ratio of the concentrations of the basic substances.

The arrangement from Fig. 4 is trained using the training data record.

In order to illustrate better the transformations achieved by means of the arrangement, steps of a method sequence 300 are illustrated in Fig. 3 with reference to the first exemplary embodiment.

In a first step 310, the arrangement is supplied with the time series values of the input variable u_t which contains the information relating to the temperature, the pressure and the agitation frequency in the reactor 100.

In a second step 320, the input variable u_t is used to determine an output variable u_t of the first neural network, which output variable u_t describes whether the basic substances are mixed together in the ratio which is optimized for an effective reaction of the basic substances and which constitutes what is referred to as an equilibrium state in the reactor. In this way, it is determined in the second step 320 whether a state in the reactor is in an equilibrium state or inequilibrium state.

In a third layer 330, the output variable s_t of the first neural network is supplied as an input variable to the second neural network.

In a fourth step 340, the second neural network determines the output variable y_t which describes a dynamic change in the concentrations of the basic substances.

In the fourth step 340, a state with which the inequilibrium state can be compensated is determined in the reactor using the determined change in the concentrations of the basic substances.

The arrangement in <u>Fig.4</u> which is trained in accordance with the training method described above is used for the open-loop and closed-loop control of the injection process of the basic substances 103 into the chemical reactor 100.

The aim of the closed-loop and open-loop control is an automated continuous injection of the basic substances 103

into the chemical reactor 100 in such a way that the concentration ratio of the basic substances 103 in the reactor 100 has a constant ratio which is optimum for the reaction to be carried out. An efficient generation of the predefinable end product as a substance 104 which flows out is thus possible.

For this purpose, a forecast value y_t is determined in an application phase by the arrangement for a first input variable y_{t-1} at a point in time t-1 and for a second input variable u_t at a point in time t, said forecast value u_t being subsequently fed as control variable 420, after possible conditioning of the determined value, to the injection device 150 in order to control the injection of the basic substances 103 in the chemical reactor 100 (cf. Fig. 1).

The structuring of the arrangement in the first neural network and the second neural network, in particular an output layer which is formed by this structuring and which produces further fault signals, ensures that a small amount of training data is required to train the arrangement in order to ensure sufficient precision during modeling of the dynamic system.

2nd Exemplary embodiment: Exchange Rate Forecast

In a second exemplary embodiment, the arrangement described above according to Fig. 4 is used for an exchange rate forecast of a $/DM exchange rate.

A time series with time series values which each comprise information on economic figures, for example short-term and long-term interest rates in a $ area and a DM area, inflation rates and information relating to economic growth in the

$ area and the DM area is fed to the arrangement as input variable u_t.

A change in the $/DM exchange rate is forecast by the arrangement as output variable y_t.

In the arrangement for the exchange rate forecast, in particular the connection matrices A, B, C and D have a particular configuration.

The first connection matrix A is filled with weightings in such a way that only a limited number of neurons of the first input layer of the first neural network (a maximum of seven neurons in this case) are assigned in each case to a neuron of the first output layer. Such a connection matrix is referred to as a "sparse connector".

In addition, the arrangement for the exchange rate forecast is configured in such a way that the first output layer of the first neural network has a large number of output neurons (200 output neurons in this case).

The second connection matrix B, in this case a highly dimensioned vector, is configured in such a way that the highly dimensional output variable s_t of the first neural network (dimension = 200) is mapped onto a one-dimensional variable using the connection matrix B. In particular, all the weightings of the second connection matrix B have the value 1.

Accordingly, the third connection matrix C only has a weighting value which additionally can assume only positive values.

The fourth connection matrix D is configured as a diagonal matrix in which all the diagonal values have the value 1.

The back propagation method is also used as the training method. A training data record is formed in the following way.

Known exchange rate changes are grouped as time series values u_t in chronological succession together with the corresponding economic figures which correspond to the known exchange rate changes.

When the arrangement is trained, these time sequence values u_t are supplied to the arrangement as a training data record.

During the training, a target function E, which is formed in the first output layer of the first neural network, has the following rule:

$$E = \frac{1}{T} \sum_t \ln \left(\frac{p_{t+1}}{p_t} \right) * z_t^a \rightarrow \max \qquad (5)$$

where:

t : index which describes a point in time

T : time interval under consideration

a : index for a neuron

ln(...) : natural logarithm

p_t, p_{t+1} : exchange rate at the point in time t or t+1

z_t^a : an exchanged quantity of money in DM which is assigned to a neuron a.

In addition, the following relationships which describe a dynamic exchange rate system apply:

$m_{t+1}^a = m_t^a - z_t^a$ (market mechanism of the exchange rate system) (6)

$n_{t+1}^a = n_t^a - p_t * z_t^a$ (market mechanism of the exchange rate system) (7)

$\sum_a z_t^a = 0$ (equilibrium condition of the exchange rate system) (8)

where:

m^a_{t+1}, m^a_t : a quantity of money in \$, assigned to a neuron a, at a point in time t+1 or t

n^a_{t+1}, n^a_t : : a quantity of money in DM, assigned to a neuron a, at a point in time t+1 or t.

The arrangement for the exchange rate forecast is based on the following principles:

An inequilibrium state of the exchange rate system which is caused by an excess of an amount of money is compensated by a change in the exchange rate.

A state of the exchange rate system is described by a decision model which is implemented in the first neural network. In this case, in each case a neuron represents what is referred to as an "agent". Accordingly, the arrangement for the exchange rate forecast is also referred to as "multi-agent system".

In the second neural network, a market model is implemented with which an inequilibrium state which is generated by the decision model is compensated by a change in a state of an exchange rate market.

An inequilibrium state of the decision model is a cause of a change in state of the market model.

An alternative to the second exemplary embodiment is described below.

The alternative exemplary embodiment differs from the second exemplary embodiment in the following descriptive points. The corresponding arrangement of the alternative exemplary embodiment for the exchange rate forecast is illustrated in

each case in fig. 5 (application phase) and fig. 6 (training phase).

Neuron connections which are broken are represented by dashed lines in the figures. Closed neuron connections are illustrated by continuous lines.

The original arrangement for the exchange rate forecast according to fig. 4 can be changed to the effect that, instead of the third neural network, a fourth input layer 550 is used with input neurons which are connected to the output neurons of the second output layer 540 of the second neural network. Weightings of the connections are contained in a fifth connection matrix E. The connections are implemented in such a way that they are broken in an application phase and closed in a training phase.

In addition, the second output layer 540 of the second neural network has a feedback with which the output signals of the second output layer 540 are fed back into the second output layer 540. Weightings of the feedback are contained in a sixth connection matrix F. The feedback is implemented in such a way that it is closed in the application phase and interrupted in the training phase.

Moreover, the output neurons of the first output layer 520 of the first neural network are connected to the output neurons 540 of the second neural network. Weightings of the connections are contained in a seventh connection matrix G.

The connections between the second input layer 530 and the second output layer 540 are configured in such a way that they are closed in an application phase and interrupted in a training phase.

The fifth connection matrix E and the sixth connection matrix F is each an identity matrix Id.

each case in Fig. 5 (application phase) and Fig. 6 (training phase).

Neuron connections which are broken are represented by dashed lines in the figures. Closed neuron connections are illustrated by continuous lines.

The original arrangement for the exchange rate forecast according to Fig. 4 can be changed to the effect that, instead of the third neural network, a fourth input layer 550 is used with input neurons which are connected to the output neurons of the second output layer 540 of the second neural network. Weightings of the connections are contained in a fifth connection matrix E. The connections are implemented in such a way that they are broken in an application phase and closed in a training phase.

In addition, the second output layer 540 of the second neural network has a feedback with which the output signals of the second output layer 540 are fed back into the second output layer 540. Weightings of the feedback are contained in a sixth connection matrix F. The feedback is implemented in such a way that it is closed in the application phase and interrupted in the training phase.

Moreover, the output neurons of the first output layer 520 of the first neural network are connected to the output neurons 540 of the second neural network. Weightings of the connections are contained in a seventh connection matrix G.

The connections between the second input layer 530 and the second output layer 540 are configured in such a way that they are closed in an application phase and interrupted in a training phase.

The fifth connection matrix E and the sixth connection matrix F is each an identity matrix Id.

The seventh connection matrix G is configured in such a way that a mapping which is carried out using the seventh connection matrix G obeys the following rule:

$$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}(\sum_a z_t^a) < 0 \qquad (9)$$

with:

$\frac{d}{d(\ln(\frac{p_{t+1}}{p_t}))}$: derived to $\ln\left(\frac{P_{t+1}}{P_t}\right)$.

Training of the alternative arrangement is carried out in accordance with the second exemplary embodiment. The alternative training arrangement is illustrated in Fig. 6.

During the training, the feedback is interrupted and the connections are broken between the second input layer 530 and the second output layer 540.

In the application phase, a change in an exchange rate with which an inequilibrium state of the market can be compensated can be tapped at the second output layer of the second neural network according to an iterative process.

Within the framework of the iterative process, the following fixed point problem is solved:

$$\ln\left(\frac{P_{t+1}}{P_t}\right) n+1 = \left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right)n + \varepsilon * \left(\sum_a a\left(\left(\ln\left(\frac{P_{t+1}}{P_t}\right)\right)n\right)\right) \qquad (10)$$

$\ln(\frac{P_{t+1}}{P_t})n$ Fixed Point

where:

ε : weighting of the third connection matrix C, $\varepsilon > 0$.

N : index for an iteration step

The alternative arrangement for the exchange rate forecast is based on the following principles:

The exchange rate system is in equilibrium at every point in time t.

By changing a state of the market model, an inequilibrium state of the decision model can be prevented.

A possible implementation of the above-described second exemplary embodiment and a possible implementation of the alternative of the second exemplary embodiment are given below for the program SENN, version 2.3. The implementations each comprise three sections which each contain a program code, said codes being necessary for processing in SENN, version 2.3.

Possible implementation of the second exemplary embodiment:

1. Parameter File:

For the application phase:

```
BpNet {
  Globals {
    WtPenalty {
     sel NoPenalty
      Weigend {
        Lambda { 0.000000 }
        AutoAdapt { T }
        w0 { 1.000000 }
        DeltaLambda { 0.000001 }
        ReducFac { 0.900000 }
        Gamma { 0.900000 }
        DesiredError { 0.000000 }
      }
      WtDecay {
        Lambda { 0.010000 }
        AutoAdapt { F }
        AdaptTime { 10 }
        EpsObj { 0.001000 }
        ObjSet { Training }
        EpsilonFac { 1.000000 }
      }
      ExtWtDecay {
        Lambda { 0.001000 }
        AutoAdapt { F }
        AdaptTime { 10 }
        EpsObj { 0.001000 }
        ObjSet { Training }
        EpsilonFac { 1.000000 }
      }
      Finnoff {
        AutoAdapt { T }
        Lambda { 0.000000 }
        DeltaLambda { 0.000001 }
        ReducFac { 0.900000 }
        Gamma { 0.900000 }
        DesiredError { 0.000000 }
      }
    }
    ErrorFunc {
     sel LnCosh
      |x| {
        parameter { 0.050000 }
      }
      LnCosh {
        parameter { 2.000000 }
      }
    }
    AnySave {
      file_name { f.Globals.dat }
    }
    AnyLoad {
      file_name { f.Globals.dat }
    }
    ASCII { F }
  }
  LearnCtrl {
   sel Stochastic
    Stochastic {
      PatternSelection {
       sel Permute
        SubSample {
          Percent { 0.500000 }
```

```
      }
      ExpRandom {
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      PruneSchedule {
       sel FixSchedule
        FixSchedule {
          Limit_0 { 10 }
          Limit_1 { 10 }
          Limit_2 { 10 }
          Limit_3 { 10 }
          RepeatLast { T }
        }
        DynSchedule {
          MaxLength { 4 }
          MinimumRuns { 0 }
          Training { F }
          Validation { T }
          Generalization { F }
        }
        DivSchedule {
          Divergence { 0.100000 }
          MinEpochs { 5 }
        }
      }
      PruneAlg {
       sel FixPrune
        FixPrune {
          Perc_0 { 0.100000 }
          Perc_1 { 0.100000 }
          Perc_2 { 0.100000 }
          Perc_3 { 0.100000 }
        }
        EpsiPrune {
          DeltaEps { 0.050000 }
          StartEps { 0.050000 }
          MaxEps { 1.000000 }
          ReuseEps { F }
        }
      }
      Tracer {
        Active { F }
        Set { Validation }
        File { trace }
      }
      Active { F }
      Randomize { 0.000000 }
      PruningSet { Train.+Valid. }
      Method { S-Pruning }
    }
    StopControl {
      EpochLimit {
        Active { F }
        MaxEpoch { 60 }
      }
      MovingExpAverage {
        Active { F }
        MaxLength { 4 }
        Training { F }
        Validation { T }
        Generalization { F }
        Decay { 0.900000 }
      }
      CheckObjectiveFct {
        Active { F }
        MaxLength { 4 }
        Training { F }
        Validation { T }
        Generalization { F }
      }
      CheckDelta {
        Active { F }
```

```
            Divergence { 0.100000 }
          }
        }
        EtaCtrl {
          Mode {
            sel EtaSchedule
            EtaSchedule {
              SwitchTime { 10 }
              ReductFactor { 0.950000 }
            }
            FuzzCtrl {
              MaxDeltaObj { 0.300000 }
              MaxDelta2Obj { 0.300000 }
              MaxEtaChange { 0.020000 }
              MinEta { 0.001000 }
              MaxEta { 0.100000 }
              Smoother { 1.000000 }
            }
          }
          Active { F }
        }
        LearnAlgo {
          sel OnlineBackProp
          VarioEta {
            MinCalls { 50 }
          }
          MomentumBackProp {
            Alpha { 0.050000 }
          }
          Quickprop {
            Decay { 0.050000 }
            Mu { 2.000000 }
          }
        }
        AnySave {
          file_name { f.Stochastic.dat }
        }
        AnyLoad {
          file_name { f.Stochastic.dat }
        }
        BatchSize { 1 }
        Eta { 0.001000 }
        DerivEps { 0.000000 }
      }
      TrueBatch {
        PatternSelection {
          sel Sequential
          SubSample {
            Percent { 0.500000 }
          }
          ExpRandom {
            Lambda { 2.000000 }
          }
        }
        WtPruneCtrl {
          Tracer {
            Active { F }
            Set { Validation }
            File { trace }
          }
          Active { F }
          Randomize { 0.000000 }
          PruningSet { Train.+Valid. }
          Method { S-Pruning }
        }
        EtaCtrl {
          Active { F }
        }
        LearnAlgo {
          sel VarioEta
          VarioEta {
            MinCalls { 200 }
          }
          MomentumBackProp {
```

```
      Alpha { 0.050000 }
    }
    Quickprop {
      Decay { 0.050000 }
      Mu { 2.000000 }
    }
  }
  AnySave {
    file_name { f.TrueBatch.dat }
  }
  AnyLoad {
    file_name { f.TrueBatch.dat }
  }
  Eta { 0.050000 }
  DerivEps { 0.000000 }
}
LineSearch {
  PatternSelection {
   sel Sequential
    SubSample {
      Percent { 0.500000 }
    }
    ExpRandom {
      Lambda { 2.000000 }
    }
  }
  WtPruneCtrl {
    Tracer {
      Active { F }
      Set { Validation }
      File { trace }
    }
    Active { F }
    Randomize { 0.000000 }
    PruningSet { Train.+Valid. }
    Method { S-Pruning }
  }
  LearnAlgo {
   sel ConjGradient
    VarioEta {
      MinCalls { 200 }
    }
    MomentumBackProp {
      Alpha { 0.050000 }
    }
    Quickprop {
      Decay { 0.050000 }
      Mu { 2.000000 }
    }
    Low-Memory-BFGS {
      Limit { 2 }
    }
  }
  AnySave {
    file_name { f.LineSearch.dat }
  }
  AnyLoad {
    file_name { f.LineSearch.dat }
  }
  EtaNull { 1.000000 }
  MaxSteps { 10 }
  LS_Precision { 0.500000 }
  TrustRegion { T }
  DerivEps { 0.000000 }
  BatchSize { 2147483647 }
}
GeneticWeightSelect {
  PatternSelection {
   sel Sequential
    SubSample {
      Percent { 0.500000 }
    }
    ExpRandom {
      Lambda { 2.000000 }
```

```
      }
    }
    LearnAlgo {
     sel VarioEta
      VarioEta {
        MinCalls { 200 }
      }
      MomentumBackProp {
        Alpha { 0.050000 }
      }
    }
    ObjFctTracer {
      Active { F }
      File { objFunc }
    }
    SearchControl {
      SearchStrategy {
       sel HillClimberControl
        HillClimberControl {
          %InitialAlive { 0.950000 }
          InheritWeights { T }
          Beta { 0.100000 }
          MutationType { DistributedMacroMutation }
          MaxTrials { 50 }
        }
        PBILControl {
          %InitialAlive { 0.950000 }
          InheritWeights { T }
          Beta { 0.100000 }
          Alpha { 0.100000 }
          PopulationSize { 40 }
        }
        PopulationControl {
          pCrossover { 1.000000 }
          CrossoverType { SimpleCrossover }
          Scaling { T }
          ScalingFactor { 2.000000 }
          Sharing { T }
          SharingFactor { 0.050000 }
          PopulationSize { 50 }
          min.%InitialAlive { 0.010000 }
          max.%InitialAlive { 0.100000 }
        }
      }
      pMutation { 0.000000 }
    }
    ObjectiveFunctionWeights {
      %Alive { 0.600000 }
      E(TS) { 0.200000 }
      Improvement(TS) { 0.000000 }
      E(VS) { 1.000000 }
      Improvement(VS) { 0.000000 }
      (E(TS)-E(VS))/max(E(TS),E(VS)) { 0.000000 }
      LipComplexity { 0.000000 }
      OptComplexity { 2.000000 }
      testVal(dead)-testVal(alive) { 0.000000 }
    }
    AnySave {
      file_name { f.GeneticWeightSelect.dat }
    }
    AnyLoad {
      file_name { f.GeneticWeightSelect.dat }
    }
    Eta { 0.050000 }
    DerivEps { 0.000000 }
    BatchSize { 5 }
    #minEpochsForFitnessTest { 2 }
    #maxEpochsForFitnessTest { 3 }
    SelectWeights { T }
    SelectNodes { T }
    maxGrowthOfValError { 0.005000 }
  }
}
CCMenu {
```

```
Clusters {
  mlp.input {
    ActFunction {
     sel id
      plogistic {
        parameter { 0.500000 }
      }
      ptanh {
        parameter { 0.500000 }
      }
      pid {
        parameter { 0.500000 }
      }
    }
    InputModification {
     sel None
      AdaptiveUniformNoise {
        NoiseEta { 1.000000 }
        DampingFactor { 1.000000 }
      }
      AdaptiveGaussNoise {
        NoiseEta { 1.000000 }
        DampingFactor { 1.000000 }
      }
      FixedUniformNoise {
        SetNoiseLevel {
          NewNoiseLevel { 1.045229 }
        }
      }
      FixedGaussNoise {
        SetNoiseLevel {
          NewNoiseLevel { 1.045229 }
        }
      }
    }
    SaveNoiseLevel {
      Filename { noise_level.dat }
    }
    LoadNoiseLevel {
      Filename { noise_level.dat }
    }
    SaveManipulatorData {
      Filename { inputManip.dat }
    }
    LoadManipulatorData {
      Filename { inputManip.dat }
    }
    Norm { NoNorm }
  }
  mlp.excessDemand {
    ActFunction {
     sel id
      plogistic {
        parameter { 0.500000 }
      }
      ptanh {
        parameter { 0.500000 }
      }
      pid {
        parameter { 0.500000 }
      }
    }
  }
  mlp.price {
    ActFunction {
     sel id
      plogistic {
        parameter { 0.500000 }
      }
      ptanh {
        parameter { 0.500000 }
      }
      pid {
        parameter { 0.500000 }
```

```
          }
        }
        ErrorFunc {
         sel LnCosh
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        ToleranceFlag { F }
        Tolerance { 0.000000 }
        Weighting { 2.000000 }
      }
      mlp.agents {
        ActFunction {
         sel tanh
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        ErrorFunc {
         sel ProfMax
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        Norm { NoNorm }
        ToleranceFlag { F }
        Tolerance { 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 }
        Weighting { 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
```

```
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000 0.100000
0.100000 0.100000 }
      }
    }
    Connectors {
      mlp.agents->excessDemand {
        WeightWatcher {
          Active { T }
          MaxWeight { 1.000000 }
          MinWeight { 1.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { F }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { F }
        EtaModifier { 1.000000 }
      }
      mlp.excessDemand->price {
        WeightWatcher {
          Active { T }
          MaxWeight { 2.000000 }
          MinWeight { 0.010000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { F }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { F }
        EtaModifier { 1.000000 }
      }
      mlp.input->agents {
        WeightWatcher {
          Active { F }
          MaxWeight { 1.000000 }
          MinWeight { 0.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { F }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { T }
        EtaModifier { 1.000000 }
      }
      mlp.bias->agents {
        WeightWatcher {
          Active { F }
          MaxWeight { 1.000000 }
          MinWeight { 0.000000 }
        }
        LoadWeightsLocal {
```

```
            Filename { std }
          }
          SaveWeightsLocal {
            Filename { std }
          }
          Alive { T }
          WtFreeze { F }
          AllowGeneticOptimization { F }
          Penalty { NoPenalty }
          AllowPruning { F }
          EtaModifier { 1.000000 }
        }
      }
      AnySave {
        file_name { f.CCMenu.dat }
      }
      AnyLoad {
        file_name { f.CCMenu.dat }
      }
    }
    TestRun {
      Filename { Test }
      Part.Transformed { F }
    }
    Online {
      Filename { Online.dat }
    }

  }
```

Für die Testphase:

```
BpNet {
  Globals {
    WtPenalty {
     sel NoPenalty
      Weigend {
        Lambda { 0.000000 }
        AutoAdapt { T }
        w0 { 1.000000 }
        DeltaLambda { 0.000001 }
        ReducFac { 0.900000 }
        Gamma { 0.900000 }
        DesiredError { 0.000000 }
      }
      WtDecay {
        Lambda { 0.010000 }
        AutoAdapt { F }
        AdaptTime { 10 }
        EpsObj { 0.001000 }
        ObjSet { Training }
        EpsilonFac { 1.000000 }
      }
      ExtWtDecay {
        Lambda { 0.001000 }
        AutoAdapt { F }
        AdaptTime { 10 }
        EpsObj { 0.001000 }
        ObjSet { Training }
        EpsilonFac { 1.000000 }
      }
      Finnoff {
        AutoAdapt { T }
        Lambda { 0.000000 }
        DeltaLambda { 0.000001 }
        ReducFac { 0.900000 }
        Gamma { 0.900000 }
        DesiredError { 0.000000 }
      }
    }
    ErrorFunc {
```

```
  sel LnCosh
   |x| {
     parameter { 0.050000 }
   }
   LnCosh {
     parameter { 2.000000 }
   }
 }
 AnySave {
   file_name { f.Globals.dat }
 }
 AnyLoad {
   file_name { f.Globals.dat }
 }
 ASCII { T }
}
LearnCtrl {
 sel Stochastic
 Stochastic {
   PatternSelection {
    sel Permute
     SubSample {
       Percent { 0.500000 }
     }
     ExpRandom {
       Lambda { 2.000000 }
     }
   }
   WtPruneCtrl {
     PruneSchedule {
      sel FixSchedule
       FixSchedule {
         Limit_0 { 10 }
         Limit_1 { 10 }
         Limit_2 { 10 }
         Limit_3 { 10 }
         RepeatLast { T }
       }
       DynSchedule {
         MaxLength { 4 }
         MinimumRuns { 0 }
         Training { F }
         Validation { T }
         Generalization { F }
       }
       DivSchedule {
         Divergence { 0.100000 }
         MinEpochs { 5 }
       }
     }
     PruneAlg {
      sel FixPrune
       FixPrune {
         Perc_0 { 0.100000 }
         Perc_1 { 0.100000 }
         Perc_2 { 0.100000 }
         Perc_3 { 0.100000 }
       }
       EpsiPrune {
         DeltaEps { 0.050000 }
         StartEps { 0.050000 }
         MaxEps { 1.000000 }
         ReuseEps { F }
       }
     }
     Tracer {
       Active { F }
       Set { Validation }
       File { trace }
     }
     Active { F }
     Randomize { 0.000000 }
     PruningSet { Train.+Valid. }
     Method { S-Pruning }
```

```
      }
      StopControl {
        EpochLimit {
          Active { F }
          MaxEpoch { 60 }
        }
        MovingExpAverage {
          Active { F }
          MaxLength { 4 }
          Training { F }
          Validation { T }
          Generalization { F }
          Decay { 0.900000 }
        }
        CheckObjectiveFct {
          Active { F }
          MaxLength { 4 }
          Training { F }
          Validation { T }
          Generalization { F }
        }
        CheckDelta {
          Active { F }
          Divergence { 0.100000 }
        }
      }
      EtaCtrl {
        Mode {
         sel EtaSchedule
          EtaSchedule {
            SwitchTime { 10 }
            ReductFactor { 0.950000 }
          }
          FuzzCtrl {
            MaxDeltaObj { 0.300000 }
            MaxDelta2Obj { 0.300000 }
            MaxEtaChange { 0.020000 }
            MinEta { 0.001000 }
            MaxEta { 0.100000 }
            Smoother { 1.000000 }
          }
        }
        Active { F }
      }
      LearnAlgo {
       sel VarioEta
        VarioEta {
          MinCalls { 50 }
        }
        MomentumBackProp {
          Alpha { 0.050000 }
        }
        Quickprop {
          Decay { 0.050000 }
          Mu { 2.000000 }
        }
      }
      AnySave {
        file_name { f.Stochastic.dat }
      }
      AnyLoad {
        file_name { f.Stochastic.dat }
      }
      BatchSize { 10 }
      Eta { 0.010000 }
      DerivEps { 0.010000 }
    }
    TrueBatch {
      PatternSelection {
       sel Sequential
        SubSample {
          Percent { 0.500000 }
        }
        ExpRandom {
```

```
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      Tracer {
        Active { F }
        Set { Validation }
        File { trace }
      }
      Active { F }
      Randomize { 0.000000 }
      PruningSet { Train.+Valid. }
      Method { S-Pruning }
    }
    EtaCtrl {
      Active { F }
    }
    LearnAlgo {
     sel VarioEta
      VarioEta {
        MinCalls { 200 }
      }
      MomentumBackProp {
        Alpha { 0.050000 }
      }
      Quickprop {
        Decay { 0.050000 }
        Mu { 2.000000 }
      }
    }
    AnySave {
      file_name { f.TrueBatch.dat }
    }
    AnyLoad {
      file_name { f.TrueBatch.dat }
    }
    Eta { 0.050000 }
    DerivEps { 0.010000 }
  }
  LineSearch {
    PatternSelection {
     sel Sequential
      SubSample {
        Percent { 0.500000 }
      }
      ExpRandom {
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      Tracer {
        Active { F }
        Set { Validation }
        File { trace }
      }
      Active { F }
      Randomize { 0.000000 }
      PruningSet { Train.+Valid. }
      Method { S-Pruning }
    }
    LearnAlgo {
     sel ConjGradient
      VarioEta {
        MinCalls { 200 }
      }
      MomentumBackProp {
        Alpha { 0.050000 }
      }
      Quickprop {
        Decay { 0.050000 }
        Mu { 2.000000 }
      }
      Low-Memory-BFGS {
        Limit { 2 }
```

```
        }
      }
      AnySave {
        file_name { f.LineSearch.dat }
      }
      AnyLoad {
        file_name { f.LineSearch.dat }
      }
      EtaNull { 1.000000 }
      MaxSteps { 10 }
      LS_Precision { 0.500000 }
      TrustRegion { T }
      DerivEps { 0.010000 }
      BatchSize { 2147483647 }
    }
    GeneticWeightSelect {
      PatternSelection {
       sel Sequential
        SubSample {
          Percent { 0.500000 }
        }
        ExpRandom {
          Lambda { 2.000000 }
        }
      }
      LearnAlgo {
       sel VarioEta
        VarioEta {
          MinCalls { 200 }
        }
        MomentumBackProp {
          Alpha { 0.050000 }
        }
      }
      ObjFctTracer {
        Active { F }
        File { objFunc }
      }
      SearchControl {
        SearchStrategy {
         sel HillClimberControl
          HillClimberControl {
            %InitialAlive { 0.950000 }
            InheritWeights { T }
            Beta { 0.100000 }
            MutationType { DistributedMacroMutation }
            MaxTrials { 50 }
          }
          PBILControl {
            %InitialAlive { 0.950000 }
            InheritWeights { T }
            Beta { 0.100000 }
            Alpha { 0.100000 }
            PopulationSize { 40 }
          }
          PopulationControl {
            pCrossover { 1.000000 }
            CrossoverType { SimpleCrossover }
            Scaling { T }
            ScalingFactor { 2.000000 }
            Sharing { T }
            SharingFactor { 0.050000 }
            PopulationSize { 50 }
            min.%InitialAlive { 0.010000 }
            max.%InitialAlive { 0.100000 }
          }
        }
        pMutation { 0.000000 }
      }
      ObjectiveFunctionWeights {
        %Alive { 0.600000 }
        E(TS) { 0.200000 }
        Improvement(TS) { 0.000000 }
        E(VS) { 1.000000 }
```

```
      Improvement(VS) { 0.000000 }
      (E(TS)-E(VS))/max(E(TS),E(VS)) { 0.000000 }
      LipComplexity { 0.000000 }
      OptComplexity { 2.000000 }
      testVal(dead)-testVal(alive) { 0.000000 }
    }
    AnySave {
      file_name { f.GeneticWeightSelect.dat }
    }
    AnyLoad {
      file_name { f.GeneticWeightSelect.dat }
    }
    Eta { 0.050000 }
    DerivEps { 0.010000 }
    BatchSize { 5 }
    #minEpochsForFitnessTest { 2 }
    #maxEpochsForFitnessTest { 3 }
    SelectWeights { T }
    SelectNodes { T }
    maxGrowthOfValError { 0.005000 }
  }
}
CCMenu {
  Clusters {
    mlp.inputP1 {
      ActFunction {
       sel id
        plogistic {
          parameter { 0.500000 }
        }
        ptanh {
          parameter { 0.500000 }
        }
        pid {
          parameter { 0.500000 }
        }
      }
      InputModification {
       sel None
        AdaptiveUniformNoise {
          NoiseEta { 1.000000 }
          DampingFactor { 1.000000 }
        }
        AdaptiveGaussNoise {
          NoiseEta { 1.000000 }
          DampingFactor { 1.000000 }
        }
        FixedUniformNoise {
          SetNoiseLevel {
            NewNoiseLevel { 0.000000 }
          }
        }
        FixedGaussNoise {
          SetNoiseLevel {
            NewNoiseLevel { 0.000000 }
          }
        }
      }
      SaveNoiseLevel {
        Filename { noise_level.dat }
      }
      LoadNoiseLevel {
        Filename { noise_level.dat }
      }
      SaveManipulatorData {
        Filename { inputManip.dat }
      }
      LoadManipulatorData {
        Filename { inputManip.dat }
      }
      Norm { NoNorm }
    }
    mlp.input {
      ActFunction {
```

```
    sel id
     plogistic {
       parameter { 0.500000 }
     }
     ptanh {
       parameter { 0.500000 }
     }
     pid {
       parameter { 0.500000 }
     }
   }
   InputModification {
    sel None
     AdaptiveUniformNoise {
       NoiseEta { 1.000000 }
       DampingFactor { 1.000000 }
     }
     AdaptiveGaussNoise {
       NoiseEta { 1.000000 }
       DampingFactor { 1.000000 }
     }
     FixedUniformNoise {
       SetNoiseLevel {
         NewNoiseLevel { 0.000000 }
       }
     }
     FixedGaussNoise {
       SetNoiseLevel {
         NewNoiseLevel { 0.000000 }
       }
     }
   }
   SaveNoiseLevel {
     Filename { noise_level.dat }
   }
   LoadNoiseLevel {
     Filename { noise_level.dat }
   }
   SaveManipulatorData {
     Filename { inputManip.dat }
   }
   LoadManipulatorData {
     Filename { inputManip.dat }
   }
   Norm { NoNorm }
 }
 mlp.agentsP1 {
   ActFunction {
    sel id
     plogistic {
       parameter { 0.500000 }
     }
     ptanh {
       parameter { 0.500000 }
     }
     pid {
       parameter { 0.500000 }
     }
   }
   ErrorFunc {
    sel none
     |x| {
       parameter { 0.050000 }
     }
     LnCosh {
       parameter { 2.000000 }
     }
   }
   Norm { NoNorm }
   ToleranceFlag { F }
   Tolerance { 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

```
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 }
        Weighting { 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 }
      }
      mlp.agents {
        ActFunction {
         sel tanh
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        ErrorFunc {
         sel ProfMax
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        Norm { NoNorm }
        ToleranceFlag { F }
        Tolerance { 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

```
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 }
        Weighting { 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 }
      }
      mlp.excessDemand {
        ActFunction {
         sel id
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
      }
      mlp.price {
        ActFunction {
         sel id
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        ErrorFunc {
         sel LnCosh
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        ToleranceFlag { F }
        Tolerance { 0.000000 }
        Weighting { 1.000000 }
      }
    }
    Connectors {
```

```
mlp.inputP1->agentsP1 {
  WeightWatcher {
    Active { F }
    MaxWeight { 1.000000 }
    MinWeight { 0.000000 }
  }
  LoadWeightsLocal {
    Filename { std }
  }
  SaveWeightsLocal {
    Filename { std }
  }
  Alive { T }
  WtFreeze { F }
  AllowPruning { T }
  EtaModifier { 1.000000 }
  Penalty { NoPenalty }
}
mlp.bias->agentsP1 {
  WeightWatcher {
    Active { F }
    MaxWeight { 1.000000 }
    MinWeight { 0.000000 }
  }
  LoadWeightsLocal {
    Filename { std }
  }
  SaveWeightsLocal {
    Filename { std }
  }
  Alive { T }
  WtFreeze { F }
  AllowPruning { F }
  EtaModifier { 1.000000 }
  Penalty { NoPenalty }
}
mlp.input->agents {
  LoadWeightsLocal {
    Filename { std }
  }
  SaveWeightsLocal {
    Filename { std }
  }
  Alive { T }
  WtFreeze { F }
  AllowPruning { T }
  EtaModifier { 1.000000 }
  Penalty { NoPenalty }
}
mlp.bias->agents {
  LoadWeightsLocal {
    Filename { std }
  }
  SaveWeightsLocal {
    Filename { std }
  }
  Alive { T }
  WtFreeze { F }
  AllowPruning { F }
  EtaModifier { 1.000000 }
  Penalty { NoPenalty }
}
mlp.agentsP1->agents {
  WeightWatcher {
    Active { F }
    MaxWeight { 1.000000 }
    MinWeight { 0.000000 }
  }
  LoadWeightsLocal {
    Filename { std }
  }
  SaveWeightsLocal {
    Filename { std }
  }
```

```
        Alive { T }
        WtFreeze { F }
        AllowPruning { F }
        Penalty { NoPenalty }
        EtaModifier { 1.000000 }
      }
      mlp.agents->excessDemand {
        WeightWatcher {
          Active { T }
          MaxWeight { 1.000000 }
          MinWeight { 1.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { T }
        AllowPruning { F }
        Penalty { NoPenalty }
        EtaModifier { 1.000000 }
      }
      mlp.excessDemand->price {
        WeightWatcher {
          Active { T }
          MaxWeight { 2.000000 }
          MinWeight { 0.010000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { F }
        AllowPruning { F }
        Penalty { NoPenalty }
        EtaModifier { 1.000000 }
      }
    }
    AnySave {
      file_name { f.CCMenu.dat }
    }
    AnyLoad {
      file_name { f.CCMenu.dat }
    }
  }
  TestRun {
    Filename { Test }
    Part.Transformed { F }
  }
  Online {
    Filename { Online.dat }
  }
}
```

2. Spezifikations-Datei:

```
APPLICATION Dollar_Prognose_Grimmdaten

MODE DAY WEEK 5

FROM 01.01.1991 TO MAX

TRAINING FROM 01.01.1991 TO 03.09.1996

VALIDATION FROM 03.09.1995 TO 03.09.1996
```

```
// VALIDATION RANDOM 0%

INPUT CLUSTER mlp.inputP1

  BEGIN  DEMUSD "DMARKER/USDOLLR"
     x = FILE data/dol.txt COLUMN 1

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  JPYUSD "JAPAYEN/USDOLLR"
     x = FILE data/dol.txt COLUMN 2

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  ECUS3M "EURO-CURRENCY (LDN) US$ 3 MONTHS - MIDDLE RATE"
     x = FILE data/dol.txt COLUMN 3

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  ECWGM3M "EURO-CURRENCY (LDN) D-MARK 3 MONTHS - MIDDLE RATE"
     x = FILE data/dol.txt COLUMN 4

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  AUSGVG4RY "US TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
     x = FILE data/dol.txt COLUMN 5

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  ABDGVG4RY "BD TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
     x = FILE data/dol.txt COLUMN 6

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END

  BEGIN  AJPGVG4RY "JP TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
     x = FILE data/dol.txt COLUMN 7

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END
```

```
  BEGIN  TOTMKUSRI "US-DS MARKET - TOT RETURN IND"
     x = FILE data/dol.txt COLUMN 8

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END


  BEGIN  TOTMKBDRI "GERMANY-DS MARKET - TOT RETURN IND"
     x = FILE data/dol.txt COLUMN 9

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END


  BEGIN  NYFECRB "COMMODITY RESEARCH BUREAU INDEX-CRB - PRICE INDEX"
     x = FILE data/dol.txt COLUMN 10

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END


  BEGIN  GOLDBLN "GOLD BULLION $/TROY OUNCE"
     x = FILE data/dol.txt COLUMN 11

     INPUT = scale((x - x(-1)) / x(-1))
             LAG -1
     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
             LAG -1
  END



INPUT CLUSTER mlp.input

  BEGIN  DEMUSD "DMARKER/USDOLLR"
     x = FILE data/dol.txt COLUMN 1

     INPUT = scale((x - x(-1)) / x(-1))

     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
  END


  BEGIN  JPYUSD "JAPAYEN/USDOLLR"
     x = FILE data/dol.txt COLUMN 2

     INPUT = scale((x - x(-1)) / x(-1))

     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
  END


  BEGIN  ECUS3M "EURO-CURRENCY (LDN) US$ 3 MONTHS - MIDDLE RATE"
     x = FILE data/dol.txt COLUMN 3

     INPUT = scale((x - x(-1)) / x(-1))

     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
  END


  BEGIN  ECWGM3M "EURO-CURRENCY (LDN) D-MARK 3 MONTHS - MIDDLE RATE"
```

```
   x = FILE data/dol.txt COLUMN 4

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  AUSGVG4RY "US TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 5

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  ABDGVG4RY "BD TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 6

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  AJPGVG4RY "JP TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 7

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  TOTMKUSRI "US-DS MARKET - TOT RETURN IND"
   x = FILE data/dol.txt COLUMN 8

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  TOTMKBDRI "GERMANY-DS MARKET - TOT RETURN IND"
   x = FILE data/dol.txt COLUMN 9

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  NYFECRB "COMMODITY RESEARCH BUREAU INDEX-CRB - PRICE INDEX"
   x = FILE data/dol.txt COLUMN 10

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END

BEGIN  GOLDBLN "GOLD BULLION $/TROY OUNCE"
   x = FILE data/dol.txt COLUMN 11

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END
```

```
TARGET CLUSTER mlp.agentsP1

        BEGIN  agents behavior past 1
             x = FILE data/dol.txt COLUMN 1

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))


          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))

          TARGET = 100 * ln(x / x(-1))
          TARGET = 100 * ln(x / x(-1))
```

```
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
```

```
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))

TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
TARGET = 100 * ln(x / x(-1))
```

```
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
    TARGET = 100 * ln(x / x(-1))
  END


TARGET CLUSTER mlp.agents

  BEGIN  agents behavior
       x = FILE data/dol.txt COLUMN 1

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
```

```
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
```

```
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
```

```
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)

    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
    TARGET = 100 * ln(x(1) / x)
  END



TARGET CLUSTER mlp.price

  BEGIN  price
         x = FILE data/dol.txt COLUMN 1

    TARGET = 100 * ln(x(1) / x)
              ASSIGN TO channel
  END


SIGNAL

  BEGIN hit rate = NORMSUM(signal)
         t =  TARGET channel
         o =  OUTPUT channel

         SIGNAL = IF t * o > 0 THEN 1 ELSE 0
  END


  BEGIN RoI
     y = FILE data/dol.txt COLUMN 1
     o = OUTPUT channel

     SIGNAL = (y(1) / y - 1)   * sign(o)
  END


  BEGIN realized potential = Relsum(signal1, signal2)
         y = FILE data/dol.txt COLUMN 1
         o = OUTPUT channel

     SIGNAL = (y(1) / y - 1)   * sign(o)

         SIGNAL = abs(y(1) / y - 1)
  END


  BEGIN  Backtransformation of forecasts
         y = FILE data/dol.txt COLUMN 1
     o = OUTPUT channel

     SIGNAL = y(1)

     SIGNAL = y * (1 + o / 100)
  END


  BEGIN Buy & Hold
         y = FILE data/dol.txt COLUMN 1
```

```
        SIGNAL = y(1) / y - 1
    END

    BEGIN Naiv Prognose
        y = FILE data/dol.txt COLUMN 1

        SIGNAL = (y(1) / y - 1) * sign(y - y(-1))
    END
```

3. Modell-Top-Datei:

```
net {
    cluster INPUT  ( EQUIVALENT, IN  );
    cluster AGENTS ( EQUIVALENT, OUT );

    connect INPUT_AGENTS ( INPUT -> AGENTS, RANDOM(20));
    connect BIAS_AGENTS  ( bias  -> AGENTS );

    INPUT  inputP1;
    INPUT  input;
    AGENTS agentsP1;
    AGENTS agents;

    cluster ( DIM(1), HID) excessDemand;
    cluster (         OUT) price;

    connect ( inputP1      -> agentsP1, INPUT_AGENTS );
    connect ( bias         -> agentsP1, BIAS_AGENTS  );
    connect ( input        -> agents  , INPUT_AGENTS );
    connect ( bias         -> agents  , BIAS_AGENTS  );
    connect ( agentsP1     -> agents  , DIAGONAL(1.0));
    connect ( agents       -> excessDemand           );
    connect ( excessDemand -> price                  );

  } mlp;
```

Mögliche Realisierung der Alternative des zweiten Ausführungsbeispiels:

1. Parameter-Datei:

```
BpNet {
  Globals {
    WtPenalty {
     sel NoPenalty
      Weigend {
        Lambda { 0.000000 }
        AutoAdapt { T }
        w0 { 1.000000 }
        DeltaLambda { 0.000001 }
        ReducFac { 0.900000 }
        Gamma { 0.900000 }
        DesiredError { 0.000000 }
      }
      WtDecay {
        Lambda { 0.010000 }
        AutoAdapt { F }
        AdaptTime { 10 }
        EpsObj { 0.001000 }
        ObjSet { Training }
        EpsilonFac { 1.000000 }
      }
      ExtWtDecay {
        Lambda { 0.001000 }
        AutoAdapt { F }
        AdaptTime { 10 }
```

```
      EpsObj { 0.001000 }
      ObjSet { Training }
      EpsilonFac { 1.000000 }
    }
    Finnoff {
      AutoAdapt { T }
      Lambda { 0.000000 }
      DeltaLambda { 0.000001 }
      ReducFac { 0.900000 }
      Gamma { 0.900000 }
      DesiredError { 0.000000 }
    }
  }
  ErrorFunc {
   sel LnCosh
    |x| {
      parameter { 0.050000 }
    }
    LnCosh {
      parameter { 2.000000 }
    }
  }
  AnySave {
    file_name { f.Globals.dat }
  }
  AnyLoad {
    file_name { f.Globals.dat }
  }
  ASCII { T }
}
LearnCtrl {
 sel Stochastic
  Stochastic {
    PatternSelection {
     sel Permute
      SubSample {
        Percent { 0.500000 }
      }
      ExpRandom {
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      PruneSchedule {
       sel FixSchedule
        FixSchedule {
          Limit_0 { 10 }
          Limit_1 { 10 }
          Limit_2 { 10 }
          Limit_3 { 10 }
          RepeatLast { T }
        }
        DynSchedule {
          MaxLength { 4 }
          MinimumRuns { 0 }
          Training { F }
          Validation { T }
          Generalization { F }
        }
        DivSchedule {
          Divergence { 0.100000 }
          MinEpochs { 5 }
        }
      }
      PruneAlg {
       sel FixPrune
        FixPrune {
          Perc_0 { 0.100000 }
          Perc_1 { 0.100000 }
          Perc_2 { 0.100000 }
          Perc_3 { 0.100000 }
        }
        EpsiPrune {
          DeltaEps { 0.050000 }
```

```
        StartEps { 0.050000 }
        MaxEps { 1.000000 }
        ReuseEps { F }
      }
    }
    Tracer {
      Active { F }
      Set { Validation }
      File { trace }
    }
    Active { F }
    Randomize { 0.000000 }
    PruningSet { Train.+Valid. }
    Method { S-Pruning }
  }
  StopControl {
    EpochLimit {
      Active { F }
      MaxEpoch { 60 }
    }
    MovingExpAverage {
      Active { F }
      MaxLength { 4 }
      Training { F }
      Validation { T }
      Generalization { F }
      Decay { 0.900000 }
    }
    CheckObjectiveFct {
      Active { F }
      MaxLength { 4 }
      Training { F }
      Validation { T }
      Generalization { F }
    }
    CheckDelta {
      Active { F }
      Divergence { 0.100000 }
    }
  }
  EtaCtrl {
    Mode {
     sel EtaSchedule
      EtaSchedule {
        SwitchTime { 10 }
        ReductFactor { 0.950000 }
      }
      FuzzCtrl {
        MaxDeltaObj { 0.300000 }
        MaxDelta2Obj { 0.300000 }
        MaxEtaChange { 0.020000 }
        MinEta { 0.001000 }
        MaxEta { 0.100000 }
        Smoother { 1.000000 }
      }
    }
    Active { F }
  }
  LearnAlgo {
   sel VarioEta
    VarioEta {
      MinCalls { 50 }
    }
    MomentumBackProp {
      Alpha { 0.050000 }
    }
    Quickprop {
      Decay { 0.050000 }
      Mu { 2.000000 }
    }
  }
  AnySave {
    file_name { f.Stochastic.dat }
  }
```

```
    AnyLoad {
      file_name { f.Stochastic.dat }
    }
    BatchSize { 10 }
    Eta { 0.010000 }
    DerivEps { 0.000000 }
  }
  TrueBatch {
    PatternSelection {
     sel Sequential
      SubSample {
        Percent { 0.500000 }
      }
      ExpRandom {
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      Tracer {
        Active { F }
        Set { Validation }
        File { trace }
      }
      Active { F }
      Randomize { 0.000000 }
      PruningSet { Train.+Valid. }
      Method { S-Pruning }
    }
    EtaCtrl {
      Active { F }
    }
    LearnAlgo {
     sel VarioEta
      VarioEta {
        MinCalls { 200 }
      }
      MomentumBackProp {
        Alpha { 0.050000 }
      }
      Quickprop {
        Decay { 0.050000 }
        Mu { 2.000000 }
      }
    }
    AnySave {
      file_name { f.TrueBatch.dat }
    }
    AnyLoad {
      file_name { f.TrueBatch.dat }
    }
    Eta { 0.050000 }
    DerivEps { 0.000000 }
  }
  LineSearch {
    PatternSelection {
     sel Sequential
      SubSample {
        Percent { 0.500000 }
      }
      ExpRandom {
        Lambda { 2.000000 }
      }
    }
    WtPruneCtrl {
      Tracer {
        Active { F }
        Set { Validation }
        File { trace }
      }
      Active { F }
      Randomize { 0.000000 }
      PruningSet { Train.+Valid. }
      Method { S-Pruning }
    }
```

```
LearnAlgo {
 sel ConjGradient
  VarioEta {
   MinCalls { 200 }
  }
  MomentumBackProp {
   Alpha { 0.050000 }
  }
  Quickprop {
   Decay { 0.050000 }
   Mu { 2.000000 }
  }
  Low-Memory-BFGS {
   Limit { 2 }
  }
}
AnySave {
  file_name { f.LineSearch.dat }
}
AnyLoad {
  file_name { f.LineSearch.dat }
}
EtaNull { 1.000000 }
MaxSteps { 10 }
LS_Precision { 0.500000 }
TrustRegion { T }
DerivEps { 0.000000 }
BatchSize { 2147483647 }
}
GeneticWeightSelect {
 PatternSelection {
  sel Sequential
   SubSample {
    Percent { 0.500000 }
   }
   ExpRandom {
    Lambda { 2.000000 }
   }
 }
 LearnAlgo {
  sel VarioEta
   VarioEta {
    MinCalls { 200 }
   }
   MomentumBackProp {
    Alpha { 0.050000 }
   }
 }
 ObjFctTracer {
   Active { F }
   File { objFunc }
 }
 SearchControl {
   SearchStrategy {
    sel HillClimberControl
     HillClimberControl {
      %InitialAlive { 0.950000 }
      InheritWeights { T }
      Beta { 0.100000 }
      MutationType { DistributedMacroMutation }
      MaxTrials { 50 }
     }
     PBILControl {
      %InitialAlive { 0.950000 }
      InheritWeights { T }
      Beta { 0.100000 }
      Alpha { 0.100000 }
      PopulationSize { 40 }
     }
     PopulationControl {
      pCrossover { 1.000000 }
      CrossoverType { SimpleCrossover }
      Scaling { T }
      ScalingFactor { 2.000000 }
```

```
            Sharing { T }
            SharingFactor { 0.050000 }
            PopulationSize { 50 }
            min.%InitialAlive { 0.010000 }
            max.%InitialAlive { 0.100000 }
          }
        }
        pMutation { 0.000000 }
      }
      ObjectiveFunctionWeights {
        %Alive { 0.600000 }
        E(TS) { 0.200000 }
        Improvement(TS) { 0.000000 }
        E(VS) { 1.000000 }
        Improvement(VS) { 0.000000 }
        (E(TS)-E(VS))/max(E(TS),E(VS)) { 0.000000 }
        LipComplexity { 0.000000 }
        OptComplexity { 2.000000 }
        testVal(dead)-testVal(alive) { 0.000000 }
      }
      AnySave {
        file_name { f.GeneticWeightSelect.dat }
      }
      AnyLoad {
        file_name { f.GeneticWeightSelect.dat }
      }
      Eta { 0.050000 }
      DerivEps { 0.000000 }
      BatchSize { 5 }
      #minEpochsForFitnessTest { 2 }
      #maxEpochsForFitnessTest { 3 }
      SelectWeights { T }
      SelectNodes { T }
      maxGrowthOfValError { 0.005000 }
    }
  }
  CCMenu {
    Clusters {
      mlp.priceInput {
        ActFunction {
          sel id
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        InputModification {
          sel None
          AdaptiveUniformNoise {
            NoiseEta { 1.000000 }
            DampingFactor { 1.000000 }
          }
          AdaptiveGaussNoise {
            NoiseEta { 1.000000 }
            DampingFactor { 1.000000 }
          }
          FixedUniformNoise {
            SetNoiseLevel {
              NewNoiseLevel { 1.045229 }
            }
          }
          FixedGaussNoise {
            SetNoiseLevel {
              NewNoiseLevel { 1.045229 }
            }
          }
        }
        SaveNoiseLevel {
          Filename { noise_level.dat }
```

```
    }
    LoadNoiseLevel {
      Filename { noise_level.dat }
    }
    SaveManipulatorData {
      Filename { inputManip.dat }
    }
    LoadManipulatorData {
      Filename { inputManip.dat }
    }
  }
  mlp.input {
    ActFunction {
     sel id
      plogistic {
        parameter { 0.500000 }
      }
      ptanh {
        parameter { 0.500000 }
      }
      pid {
        parameter { 0.500000 }
      }
    }
    InputModification {
     sel None
      AdaptiveUniformNoise {
        NoiseEta { 1.000000 }
        DampingFactor { 1.000000 }
      }
      AdaptiveGaussNoise {
        NoiseEta { 1.000000 }
        DampingFactor { 1.000000 }
      }
      FixedUniformNoise {
        SetNoiseLevel {
          NewNoiseLevel { 1.045229 }
        }
      }
      FixedGaussNoise {
        SetNoiseLevel {
          NewNoiseLevel { 1.045229 }
        }
      }
    }
    SaveNoiseLevel {
      Filename { noise_level.dat }
    }
    LoadNoiseLevel {
      Filename { noise_level.dat }
    }
    SaveManipulatorData {
      Filename { inputManip.dat }
    }
    LoadManipulatorData {
      Filename { inputManip.dat }
    }
    Norm { NoNorm }
  }
  mlp.excessDemand {
    ActFunction {
     sel id
      plogistic {
        parameter { 0.500000 }
      }
      ptanh {
        parameter { 0.500000 }
      }
      pid {
        parameter { 0.500000 }
      }
    }
    ErrorFunc {
     sel |x|
```

```
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        ToleranceFlag { F }
        Tolerance { 0.000000 }
        Weighting { 30.000000 }
      }
      mlp.agents {
        ActFunction {
         sel tanh
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        ErrorFunc {
         sel ProfMax
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        Norm { NoNorm }
        ToleranceFlag { F }
        Tolerance { 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 }
        Weighting { 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
```

```
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 }
      }
      mlp.priceOutput {
        ActFunction {
          sel id
          plogistic {
            parameter { 0.500000 }
          }
          ptanh {
            parameter { 0.500000 }
          }
          pid {
            parameter { 0.500000 }
          }
        }
        ErrorFunc {
          sel none
          |x| {
            parameter { 0.050000 }
          }
          LnCosh {
            parameter { 2.000000 }
          }
        }
        ToleranceFlag { F }
        Tolerance { 0.000000 }
        Weighting { 1.000000 }
      }
    }
    Connectors {
      mlp.agents->excessDemand {
        WeightWatcher {
          Active { T }
          MaxWeight { 1.000000 }
          MinWeight { 1.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { T }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { F }
        EtaModifier { 1.000000 }
      }
      mlp.priceOutput->agents {
        WeightWatcher {
          Active { T }
          MaxWeight { -0.001000 }
          MinWeight { -2.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { T }
        WtFreeze { F }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { F }
        EtaModifier { 1.000000 }
      }
      mlp.input->agents {
        WeightWatcher {
          Active { F }
          MaxWeight { 1.000000 }
```

```
      MinWeight { 0.000000 }
    }
    LoadWeightsLocal {
      Filename { std }
    }
    SaveWeightsLocal {
      Filename { std }
    }
    Alive { T }
    WtFreeze { F }
    AllowGeneticOptimization { F }
    Penalty { NoPenalty }
    AllowPruning { T }
    EtaModifier { 1.000000 }
  }
  mlp.bias->agents {
    WeightWatcher {
      Active { F }
      MaxWeight { 1.000000 }
      MinWeight { 0.000000 }
    }
    LoadWeightsLocal {
      Filename { std }
    }
    SaveWeightsLocal {
      Filename { std }
    }
    Alive { T }
    WtFreeze { F }
    AllowGeneticOptimization { F }
    Penalty { NoPenalty }
    AllowPruning { F }
    EtaModifier { 1.000000 }
  }
  mlp.priceInput->priceOutput {
    WeightWatcher {
      Active { F }
      MaxWeight { 1.000000 }
      MinWeight { 0.000000 }
    }
    LoadWeightsLocal {
      Filename { std }
    }
    SaveWeightsLocal {
      Filename { std }
    }
    Alive { T }
    WtFreeze { T }
    AllowGeneticOptimization { F }
    Penalty { NoPenalty }
    AllowPruning { F }
    EtaModifier { 1.000000 }
  }
  mlp.excessDemand->priceOutput {
    WeightWatcher {
      Active { T }
      MaxWeight { -0.010000 }
      MinWeight { -0.010000 }
    }
    LoadWeightsLocal {
      Filename { std }
    }
    SaveWeightsLocal {
      Filename { std }
    }
    Alive { F }
    WtFreeze { T }
    AllowGeneticOptimization { F }
    Penalty { NoPenalty }
    AllowPruning { F }
    EtaModifier { 1.000000 }
  }
  mlp.priceOutput->priceOutput {
    WeightWatcher {
```

```
          Active { F }
          MaxWeight { 1.000000 }
          MinWeight { 0.000000 }
        }
        LoadWeightsLocal {
          Filename { std }
        }
        SaveWeightsLocal {
          Filename { std }
        }
        Alive { F }
        WtFreeze { T }
        AllowGeneticOptimization { F }
        Penalty { NoPenalty }
        AllowPruning { F }
        EtaModifier { 1.000000 }
      }
    }
    AnySave {
      file_name { f.CCMenu.dat }
    }
    AnyLoad {
      file_name { f.CCMenu.dat }
    }
  }
  RecPar {
    decay_c { 1.000000 }
    delta_t { 1.000000 }
    epsilon { 0.001000 }
    max_iter { 30 }
    show { T }
    Reset_Errors { T }
  }
  TestRun {
    Filename { Test }
    Part.Transformed { F }
  }
  Online {
    Filename { Online.dat }
  }

}
```

2. Spezifikations-Datei:

```
APPLICATION Prognose
```

```
MODE DAY WEEK 5

FROM 01.01.1991 TO MAX

TRAINING FROM 01.01.1991 TO 03.09.1996

VALIDATION FROM 03.09.1995 TO 03.09.1996


INPUT

  BEGIN  DEMUSD "DMARKER/USDOLLR"
     x = FILE data/dol.txt COLUMN 1

     INPUT = scale((x - x(-1)) / x(-1))

     INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
  END


  BEGIN  JPYUSD "JAPAYEN/USDOLLR"
```

```
   x = FILE data/dol.txt COLUMN 2

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  ECUS3M "EURO-CURRENCY (LDN) US$ 3 MONTHS - MIDDLE RATE"
   x = FILE data/dol.txt COLUMN 3

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  ECWGM3M "EURO-CURRENCY (LDN) D-MARK 3 MONTHS - MIDDLE RATE"
   x = FILE data/dol.txt COLUMN 4

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  AUSGVG4RY "US TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 5

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  ABDGVG4RY "BD TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 6

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  AJPGVG4RY "JP TOTAL 7-10 YEARS DS GOVT. INDEX - RED. YIELD"
   x = FILE data/dol.txt COLUMN 7

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  TOTMKUSRI "US-DS MARKET - TOT RETURN IND"
   x = FILE data/dol.txt COLUMN 8

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  TOTMKBDRI "GERMANY-DS MARKET - TOT RETURN IND"
   x = FILE data/dol.txt COLUMN 9

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  NYFECRB "COMMODITY RESEARCH BUREAU INDEX-CRB - PRICE INDEX"
   x = FILE data/dol.txt COLUMN 10
```

```
   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


BEGIN  GOLDBLN "GOLD BULLION $/TROY OUNCE"
   x = FILE data/dol.txt COLUMN 11

   INPUT = scale((x - x(-1)) / x(-1))

   INPUT = scale((x - 2 * x(-1) + x(-2)) / x)
END


TARGET CLUSTER mlp.excessDemand

       BEGIN  excessDemand

          TARGET = 0
       END


TARGET CLUSTER mlp.agents

       BEGIN  agents behavior
              x = FILE data/dol.txt COLUMN 1

          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)

          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)

          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)

          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
          TARGET = 100 * ln(x(1) / x)
```

```
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
```

```
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)

TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
TARGET = 100 * ln(x(1) / x)
```

```
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)

        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)

        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
        TARGET = 100 * ln(x(1) / x)
      END


TARGET CLUSTER mlp.priceOutput

      BEGIN  price
             x = FILE data/dol.txt COLUMN 1

         TARGET = 100 * ln(x(1) / x)
                  ASSIGN TO channel
      END


SIGNAL

      BEGIN hit rate = NORMSUM(signal)
             t =  TARGET channel
             o =  OUTPUT channel

             SIGNAL = IF t * o > 0 THEN 1 ELSE 0
      END


      BEGIN RoI
         y = FILE data/dol.txt COLUMN 1
         o = OUTPUT channel

         SIGNAL = (y(1) / y - 1)  * sign(o)
      END


      BEGIN realized potential = Relsum(signal1, signal2)
             y = FILE data/dol.txt COLUMN 1
             o = OUTPUT channel

         SIGNAL = (y(1) / y - 1)  * sign(o)

             SIGNAL = abs(y(1) / y - 1)
      END
```

```
BEGIN  Backtransformation of forecasts
       y = FILE data/dol.txt COLUMN 1
   o = OUTPUT channel

   SIGNAL = y(1)

   SIGNAL = y * (1 + o / 100)
END


BEGIN Buy & Hold
       y = FILE data/dol.txt COLUMN 1

       SIGNAL = y(1) / y - 1
END


BEGIN Naiv Prognose
       y = FILE data/dol.txt COLUMN 1

       SIGNAL = (y(1) / y - 1) * sign(y - y(-1))
END
```

3. Modell-Top-Datei:

```
net {
        cluster ( IN  ) priceInput;
        cluster ( IN  ) input;
       cluster ( OUT ) excessDemand;
       cluster ( OUT ) agents;
       cluster ( OUT ) priceOutput;

       connect ( priceInput   -> priceOutput, 1TO1 );
       connect ( priceOutput  -> agents            );
       connect ( input        -> agents, RANDOM(32));
        connect ( bias         -> agents            );
       connect ( agents       -> excessDemand      );
        connect ( excessDemand -> priceOutput       );
        connect ( priceOutput  -> priceOutput, 1TO1 );
    } mlp;
```

In diesem Dokument sind folgende Veröffentlichungen zitiert:

[1] S. Haykin, Neural Networks: A Comprehensive Foundation, Mc Millan College Publishing Company, ISBN 0-02-352761-7, S. 498-533, 1994.

[2] A. Zell, Simulation Neuronaler Netze, Addison-Wesley Publishing Company, S.560-561, 1. Auflage, Bonn, 1994

Patent claims

1. An arrangement for the computer-supported compensation of an inequilibrium state of a first technical system,
- having a first neural network, which describes the first technical system;
- having a second neural network, which describes a second technical system;
- in which the first and the second neural network are connected to one another in such a way that an inequilibrium state of the first technical system can be compensated by the second neural network.

2. The arrangement as claimed in claim 1, in which the first neural network has at least a first input computing element and a second output computing element.

3. The arrangement as claimed in claim 1 or 2, in which the second neural network has at least a second input computing element and a second output computing element.

4. The arrangement as claimed in one of claims 1 to 3, in which at least some of the computing elements are artificial neurons.

5. The arrangement as claimed in one of claims 1 to 4, in which at least some of the connections between computing elements are of variable configuration.

6. The arrangement as claimed in one of claims 1 to 5, in which at least some of the connections have identical weighting values.

7. The arrangement as claimed in one of claims 1 to 6, in which the first technical system and the second technical system are each subsystems of a common overall system.

8. The arrangement as claimed in one of claims 1 to 6, in which the first technical system and the second technical system are identical.

9. The arrangement as claimed in claim 7 or 8, used for determining a dynamic of a system.

10. The arrangement as claimed in one of claims 7 to 9, used for predicting a future state of a system.

11. The arrangement as claimed in claim 10, used for monitoring and/or controlling a system.

12. The arrangement as claimed in claim 11, in which the system is a chemical reactor.

13. A method for the computer-supported compensation of an inequilibrium state of a first technical system,
- in which a first neural network, which describes the first technical system, is supplied with a first input variable;
- in which a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network;
- in which the first output variable is supplied, as a second input variable, to a second neural network which describes a second technical system;
- in which a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

14. The method as claimed in claim 13, in which the first technical system and the second technical system each describe a subsystem of a common overall system.

15. The method as claimed in claim 14, in which a dynamic of the overall system is determined using the state of the second technical system.

16. The method as claimed in one of claims 13 to 15, used for predicting a future state of a system.

17. The method as claimed in claim 16, used for monitoring and/or controlling a system.

18. A computer program event which comprises a computer-readable storage medium on which a program is stored which, after it has been loaded into a memory of a computer, makes it possible for the computer to execute the following steps for the computer-supported compensation of an inequilibrium state of a first technical system:
- a first neural network, which describes the first technical system, is supplied with a first input variable;
- a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network;
- the first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system;
- a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network, in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

19. A computer-readable storage medium on which a program is stored which, after it has been loaded into a memory of the computer, permits the computer to execute the

following steps for the computer-supported compensation of an inequilibrium state of a first technical system:

- a first neural network, which describes the first technical system, is supplied with a first input variable;
- a first output variable, which describes an inequilibrium state of the first technical system, is determined for the first input variable using the first neural network;
- the first output variable is supplied, as a second input variable, to a second neural network, which describes a second technical system;
- a second output variable, which describes a state of the second technical system, is determined for the second input variable using the second neural network in such a way that the inequilibrium state of the first technical system is compensated by the second neural network.

Abstract

Arrangement and method and computer program event and computer-readable storage medium for the computer-supported compensation of an inequilibrium state

In the arrangement and the method for the computer-supported compensation of an inequilibrium state of a first technical system, a first neural network describes the first technical system and a second neural network describes a second technical system. The first and the second neural network are connected to one another in such a way that an inequilibrium state of the first technical system is compensated by the second neural network.

FIG 1

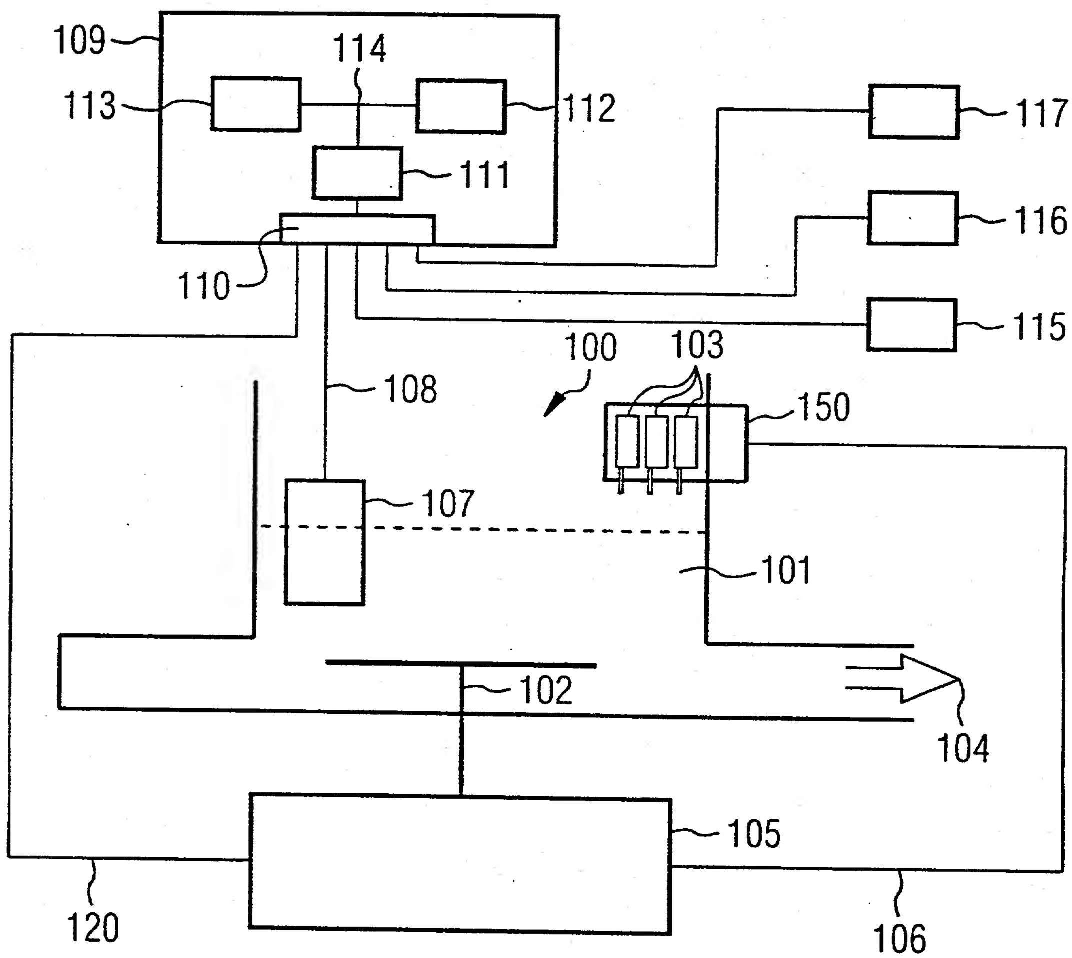

109
114
113
112
117
111
116
110
115
100
103
108
150
107
101
102
104
105
120
106

FIG 2

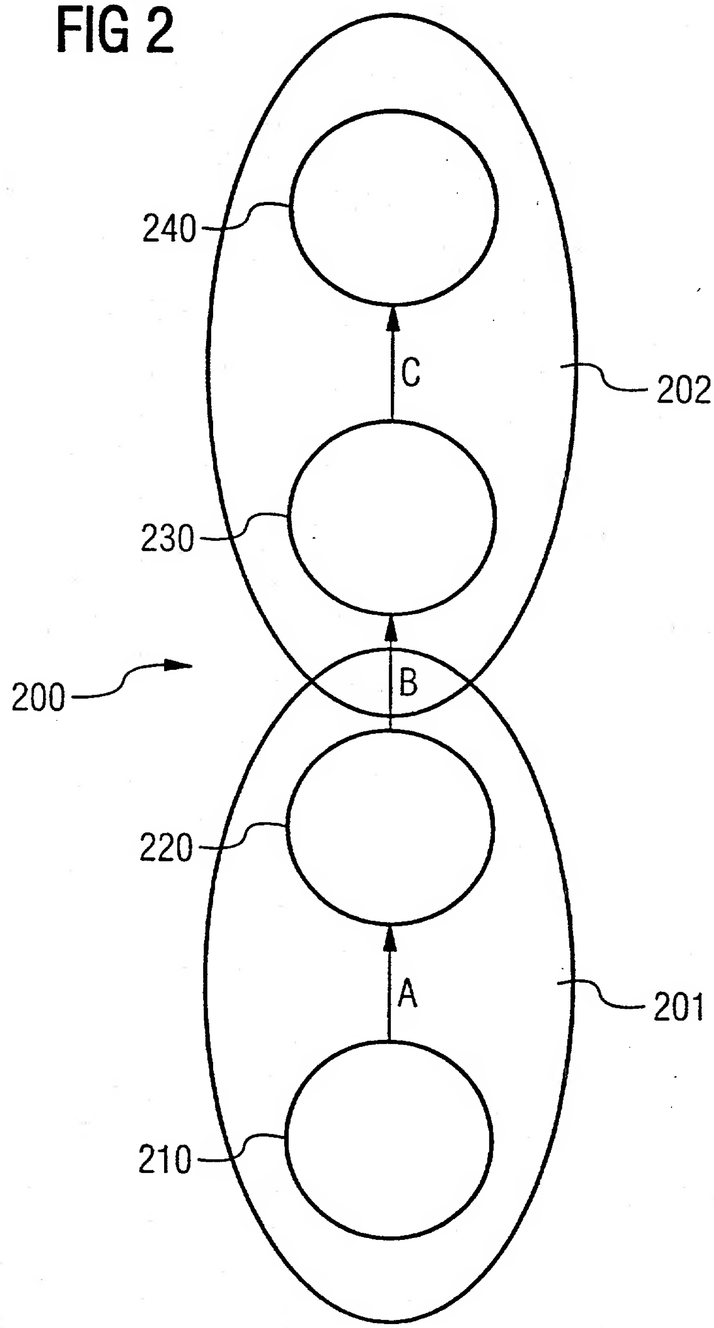


FIG 3

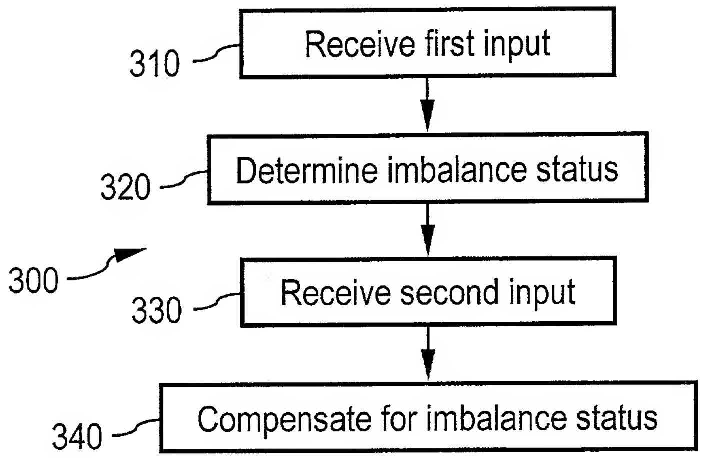

Receive first input
Determine imbalance status
Receive second input
Compensate for imbalance status
310
320
300
330
340

FIG 4

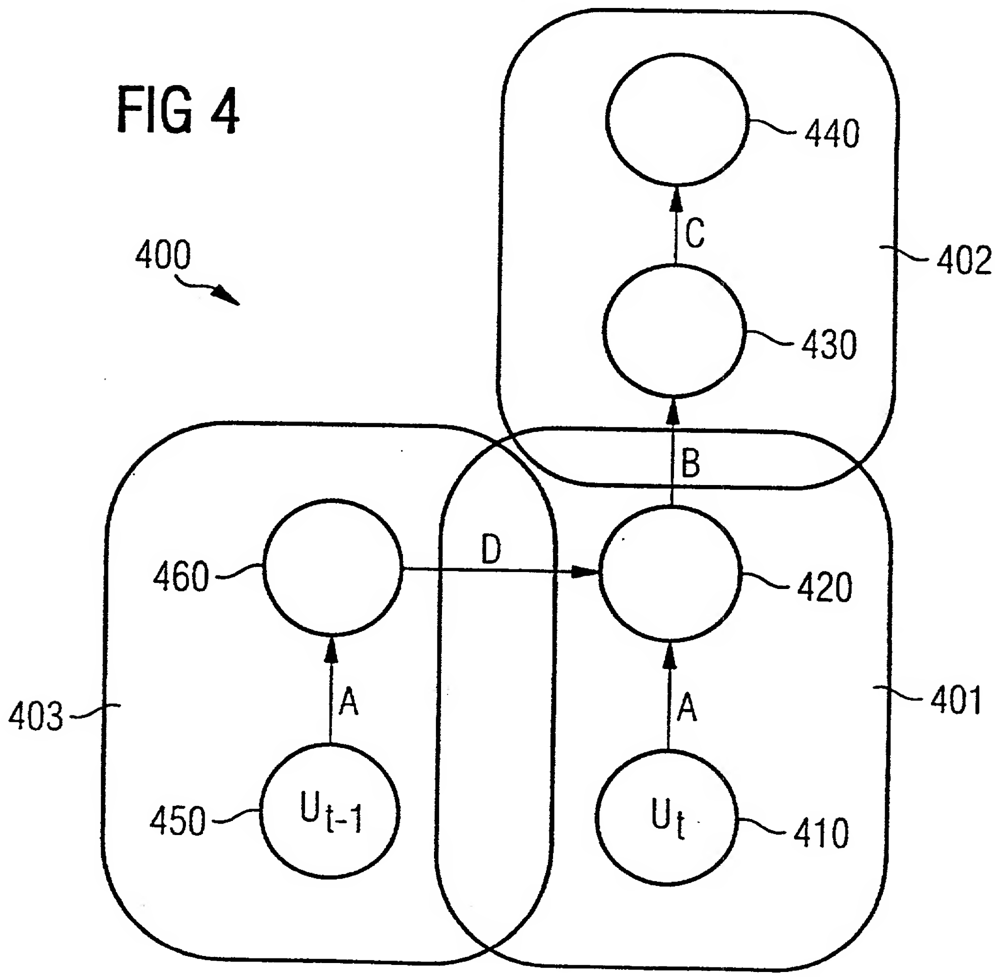

400
440
C
402
430
B
460
D
420
A
A
403
401
450
U_{t-1}
U_t
410

FIG 5

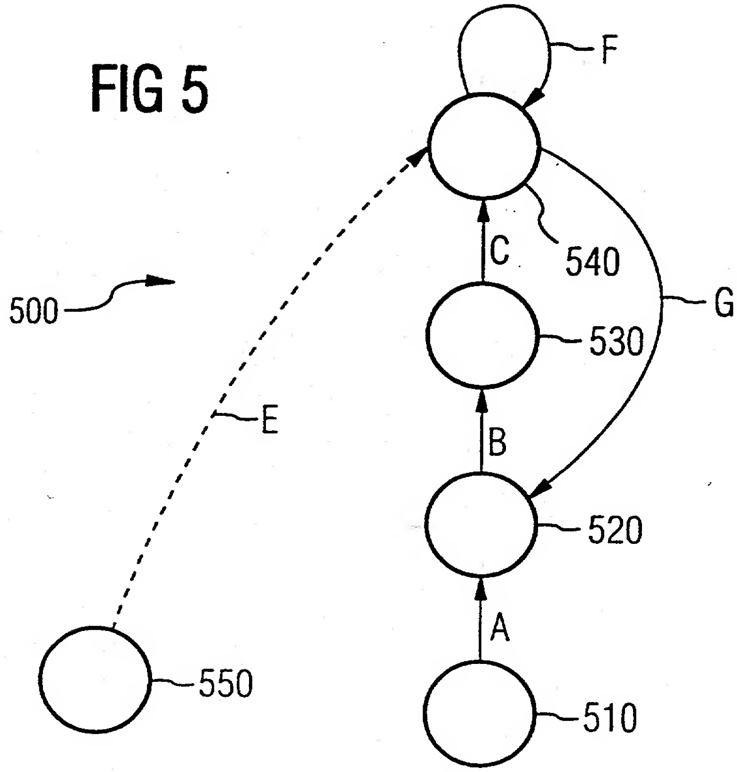


FIG 6

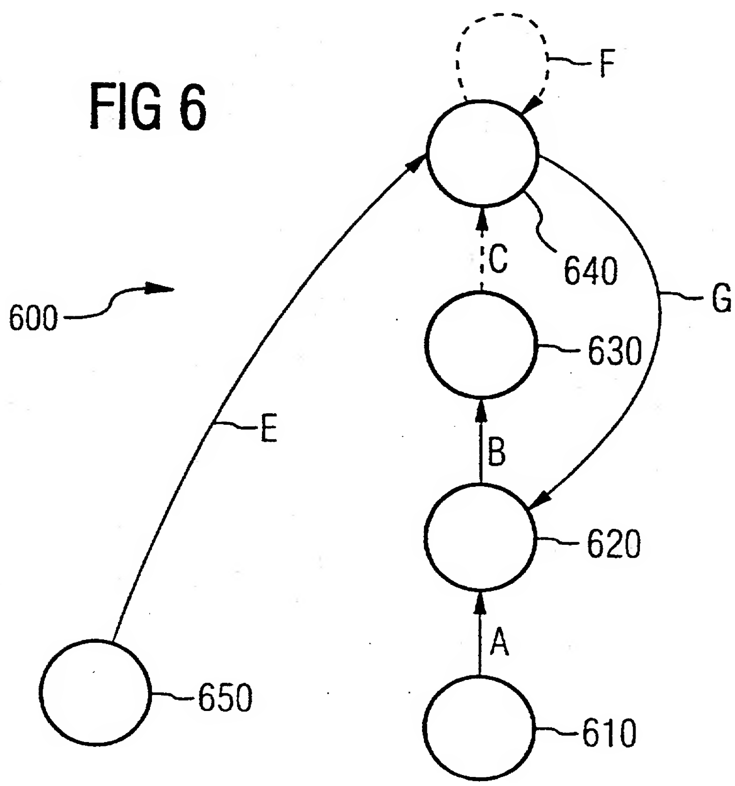
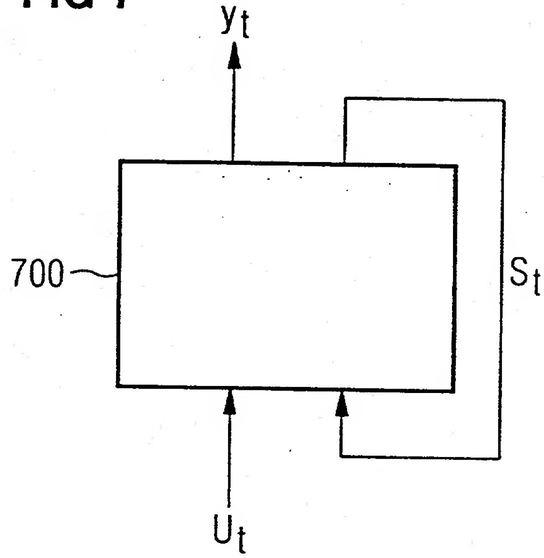


FIG 7



Declaration and Power of Attorney For Patent Application
Erklärung Für Patentanmeldungen Mit Vollmacht
German Language Declaration

Als nachstehend benannter Erfinder erkläre ich hiermit an Eides Statt:

dass mein Wohnsitz, meine Postanschrift, und meine Staatsangehörigkeit den im Nachstehenden nach meinem Namen aufgeführten Angaben entsprechen,

dass ich, nach bestem Wissen der ursprüngliche, erste und alleinige Erfinder (falls nachstehend nur ein Name angegeben ist) oder ein ursprünglicher, erster und Miterfinder (falls nachstehend mehrere Namen aufgeführt sind) des Gegenstandes bin, für den dieser Antrag gestellt wird und für den ein Patent beantragt wird für die Erfindung mit dem Titel:

Anordnung und Verfahren sowie Computerprogramm-Erzeugnis und computerlesbares Speichermedium zur rechnergestützten Kompensation eines Ungleichgewichtszustands eines technischen Systems

deren Beschreibung

(zutreffendes ankreuzen)
☐ hier beigefügt ist.
☒ am 30.05.2000 als
PCT internationale Anmeldung
PCT Anmeldungsnummer PCT/DE00/01764

eingereicht wurde und am ______
abgeändert wurde (falls tatsächlich abgeändert).

Ich bestätige hiermit, dass ich den Inhalt der obigen Patentanmeldung einschliesslich der Ansprüche durchgesehen und verstanden habe, die eventuell durch einen Zusatzantrag wie oben erwähnt abgeändert wurde.

Ich erkenne meine Pflicht zur Offenbarung irgendwelcher Informationen, die für die Prüfung der vorliegenden Anmeldung in Einklang mit Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) von Wichtigkeit sind, an.

Ich beanspruche hiermit ausländische Prioritätsvorteile gemäss Abschnitt 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 119 aller unten angegebenen Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde, und habe auch alle Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde nachstehend gekennzeichnet, die ein Anmeldedatum haben, das vor dem Anmeldedatum der Anmeldung liegt, für die Priorität beansprucht wird.

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Assembly, method, computer programme and storage medium which can be computer-read for the computer-aided compensation of a state of inequilibrium

the specification of which

(check one)
☐ is attached hereto.
☒ was filed on 30.05.2000 as
PCT international application
PCT Application No. PCT/DE00/01764
and was amended on ______
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

German Language Declaration

Prior foreign appplications
Priorität beansprucht

Priority Claimed

19928776.7 (Number) (Nummer)	DE (Country) (Land)	23.06.1999 (Day Month Year Filed) (Tag Monat Jahr eingereicht)	☒ Yes Ja	☐ No Nein
(Number) (Nummer)	(Country) (Land)	(Day Month Year Filed) (Tag Monat Jahr eingereicht)	☐ Yes Ja	☐ No Nein
(Number) (Nummer)	(Country) (Land)	(Day Month Year Filed) (Tag Monat Jahr eingereicht)	☐ Yes Ja	☐ No Nein

Ich beanspruche hiermit gemäss Absatz 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 120, den Vorzug aller unten aufgeführten Anmeldungen und falls der Gegenstand aus jedem Anspruch dieser Anmeldung nicht in einer früheren amerikanischen Patentanmeldung laut dem ersten Paragraphen des Absatzes 35 der Zivilprozeßordnung der Vereinigten Staaten, Paragraph 122 offenbart ist, erkenne ich gemäss Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) meine Pflicht zur Offenbarung von Informationen an, die zwischen dem Anmeldedatum der früheren Anmeldung und dem nationalen oder PCT internationalen Anmeldedatum dieser Anmeldung bekannt geworden sind.

I hereby claim the benefit under Title 35. United States Code. §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §122, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occured between the filing date of the prior application and the national or PCT international filing date of this application.

PCT(DE00/01764 (Application Serial No.) (Anmeldeseriennummer)	30.05.2000 (Filing Date D, M, Y) (Anmeldedatum T, M, J)	anhängig (Status) (patentiert, anhängig, aufgegeben)	pending (Status) (patented, pending, abandoned)
(Application Serial No.) (Anmeldeseriennummer)	(Filing Date D,M,Y) (Anmeldedatum T, M; J)	(Status) (patentiert, anhängig, aufgeben)	(Status) (patented, pending, abandoned)

Ich erkläre hiermit, dass alle von mir in der vorliegenden Erklärung gemachten Angaben nach meinem besten Wissen und Gewissen der vollen Wahrheit entsprechen, und dass ich diese eidesstattliche Erklärung in Kenntnis dessen abgebe, dass wissentlich und vorsätzlich falsche Angaben gemäss Paragraph 1001, Absatz 18 der Zivilprozessordnung der Vereinigten Staaten von Amerika mit Geldstrafe belegt und/oder Gefängnis bestraft werden koennen, und dass derartig wissentlich und vorsätzlich falsche Angaben die Gültigkeit der vorliegenden Patentanmeldung oder eines darauf erteilten Patentes gefährden können.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

German Language Declaration

VERTRETUNGSVOLLMACHT: Als benannter Erfinder beauftrage ich hiermit den nachstehend benannten Patentanwalt (oder die nachstehend benannten Patentanwälte) und/oder Patent-Agenten mit der Verfolgung der vorliegenden Patentanmeldung sowie mit der Abwicklung aller damit verbundenen Geschäfte vor dem Patent- und Warenzeichenamt: *(Name und Registrationsnummer anführen)*

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. *(list name and registration number)*

And I hereby appoint

Customer No. 21171

Telefongespräche bitte richten an: *(Name und Telefonnummer)*

Direct Telephone Calls to: *(name and telephone number)*

Ext. ____________

Postanschrift:

Send Correspondence to:

Staas & Halsey LLP
700 Eleventh Street NW, Suite 500 20001 Washington, DC
Telephone: (001) 202 434 1500 and Facsimile (001) 202 434 1501
or
Customer No. 21171

Voller Name des einzigen oder ursprünglichen Erfinders:	Full name of sole or first inventor:
Dr. RALF NEUNEIER	Dr. RALF NEUNEIER
Unterschrift des Erfinders / Datum: 10.12.2001	Inventor's signature / Date
Wohnsitz: MUENCHEN, DEUTSCHLAND	Residence: MUENCHEN, GERMANY DEX
Staatsangehörigkeit: DE	Citizenship: DE
Postanschrift: GRAVELOTTESTR. 3	Post Office Addess: GRAVELOTTESTR. 3
81667 MUENCHEN	81667 MUENCHEN
Voller Name des zweiten Miterfinders (falls zutreffend):	Full name of second joint inventor, if any:
Dr. HANS-GEORG ZIMMERMANN	Dr. HANS-GEORG ZIMMERMANN
Unterschrift des Erfinders / Datum: 10.12.2001	Second Inventor's signature / Date
Wohnsitz: STARNBERG/PERCHA, DEUTSCHLAND	Residence: STARNBERG/PERCHA, GERMANY DEX
Staatsangehörigkeit: DE	Citizenship: DE
Postanschrift: SCHIFFBAUERWEG 6A	Post Office Address: SCHIFFBAUERWEG 6A
82319 STARNBERG/PERCHA	82319 STARNBERG/PERCHA

(Bitte entsprechende Informationen und Unterschriften im Falle von dritten und weiteren Miterfindern angeben).

(Supply similar information and signature for third and subsequent joint inventors).